SZEREGOWANIE PROCESÓW

UNIX LINUX (>2.6) WINDOWS

WSPÓLNE CECHY ALGORYTMÓW SZEREGUJĄCYCH

- Rotacyjny z wywłaszczaniem
- Dynamiczne priorytety (wyjątek: scr – statyczne -- Linux, Windows)
- Kolejka priorytetowa (nagłówki odpowiadają priorytetom procesów)

CEL

Maksymalne wykorzystanie zasobów

(preferowane zadania ograniczone przez we/wy)

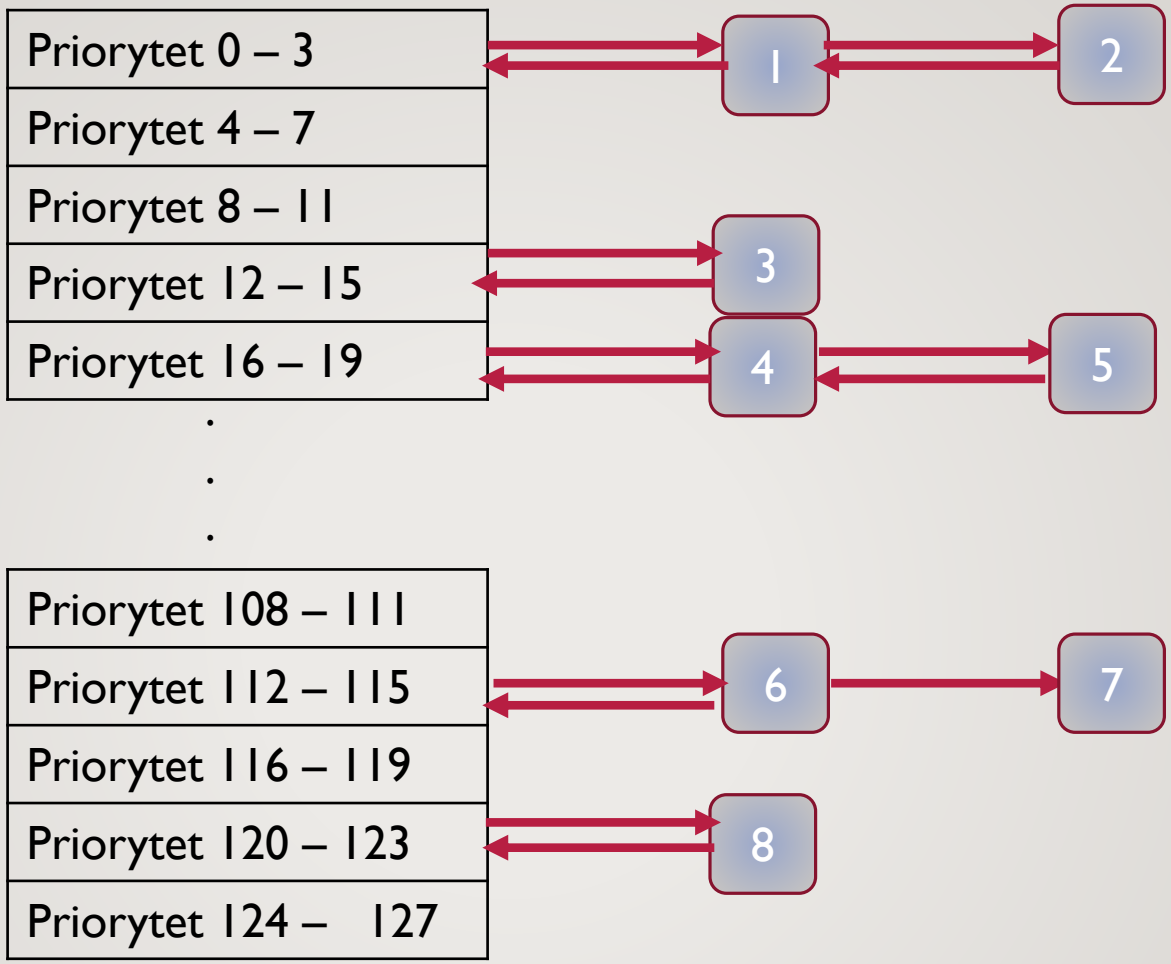
UNIX

- Priorytety <0;127> max. priorytet – 0 <0; 49> - procesy w trybie jądra
- Składowe priorytetu:
  - statyczna
    - Baza (system)
    - Nice (użytkownik)
  - modyfikowana przez planistę
- Kiedy przeliczany:
  - Proces: tryb jądra -> tryb użytkownika
  - Co zadany kwant czasu (np. 1 s)
- Algorytm rotacyjny z wywłaszczaniem ale procesy jądra niewywłaszczalne
- Przełączenie kontekstu:
  - Bieżący proces się skończył
  - Bieżący proces wchodzi w stan oczekiwania (np. op. we./wy)
  - Inny proces ma wyższy priorytet (obudzony lub w wyniku przeliczenia priorytetów)

KOLEJKA PRIORYTETOWA `QS`

- `qs` - tablica wskaźników do list dwukierunkowych
- 1 pozycja – 4 priorytety
- Niepuste pozycje kolejki – wskazane przez wektor `whichqs`
- Zakresy priorytetów:
  - Nieprzerywalne jądra
  - Przerywalne jądra
  - Użytkownika
- Obudzone procesy z trybu jądra – priorytet uśpienia (np. 20 – dla operacji dyskowej)

qs



1 – 8 – deskryptory procesów (struktury proc)

whichqs

PRZELICZANIE PRIORYTETU PROCESU

- cpu – miara wykorzystania procesora (zwiększana, gdy proces wykonywany)
- baza – priorytet bazowy
- nice (domyślnie 20)
- pri
- uspr – priorytet w trybie użytkownika

- cpu=cpu+1 - w kolejnym takcie zegara (co 1 lub 4)
- cpu=cpu/2 - przy każdym przeliczaniu priorytetów
- uspri=baza+cpu/2+nice

W trybie użytkownika pri=uspri (może się łączyć ze zmianą kolejki)

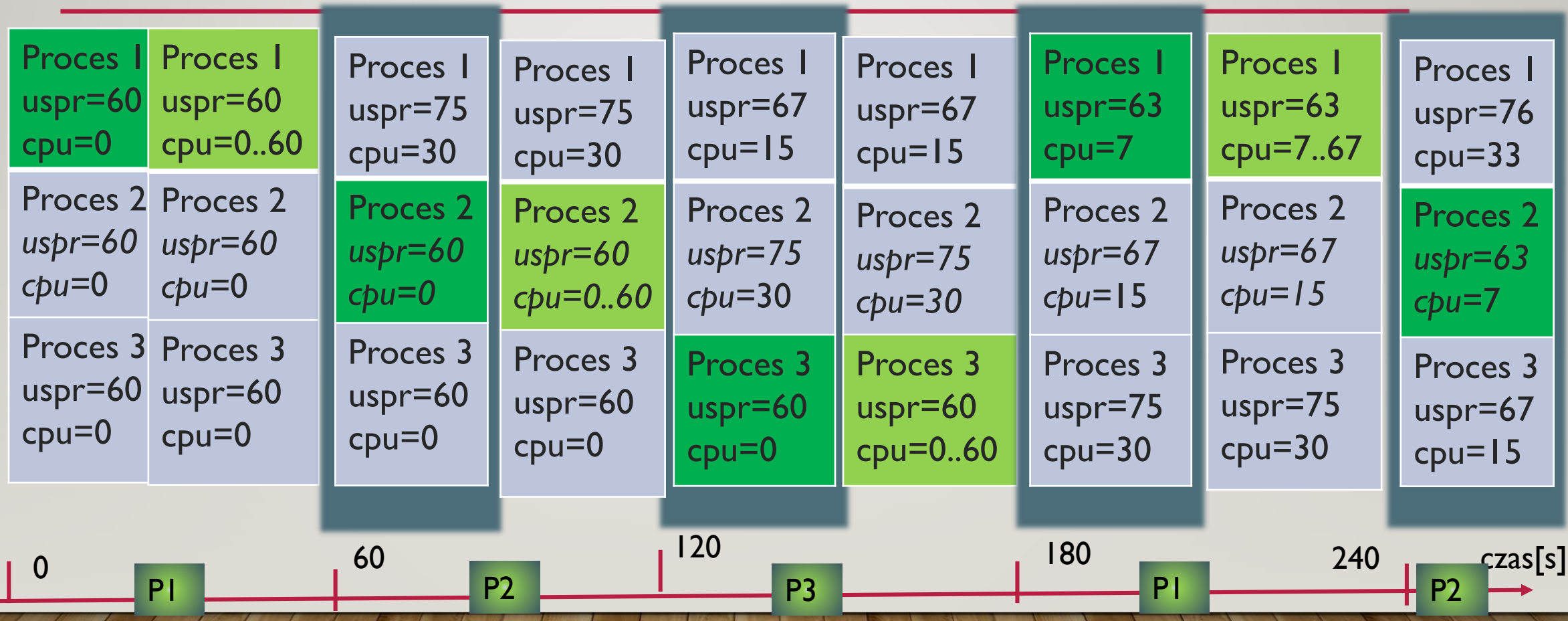
cpu=cpu+1

cpu=cpu/2

uspri=baza+cpu/2+nice

baza+nice=60 / początkowy priorytet

Przeliczanie priorytetów co 60s



LINUX (2.6)

- Priorytety: <0; 140> -- tyle rekordów w kolejce priorytetowej (tablica priorytetów)
- Zmienny kwant przydzielanego czasu
- Klasy szeregowania
  - SCHED_RR (rotacyjny real time)
  - SCHED_FIFO (fifo real time)
  - SCHED_OTHER (zwykłe zadania)
- Priorytet:
  - Część statyczna: nice <-20; 19>
  - Część dynamiczna (wpływa na kolejność i wielkość kwantu czasu)
- Priorytety <0; 99> są statyczne - dla RT(tylko root)

- Struktury danych:
  - Tablica priorytetów dla zadań aktywnych (oczekują, ale nie wykorzystały swojego czasu)
  - Tablica procesów dla zadań, które wykorzystały kwant czasu
  - Mapy bitowe dla każdej z tablic (niepuste kolejki)
- Priorytet przeliczany na bieżąco (nie co określony czas); po upływie kwantu czasu
- Wywłaszczenie (w trybie jądra ustawiony znacznik need_resched) gdy:
  - Upłynął kwant czasu dla bieżącego wykonywanego zadania
  - Zadanie o wyższym priorytecie jest gotowe do wykonania

- Po upływie kwantu czasu – obliczany priorytet i następny kwant (często bez zmian)
- Proces może trafić do kolejki aktywnych lub przeterminowanych zadań (tam czeka do końca epoki)
- Jeśli proces tworzy proces potomny pozostały kwant dzielony na pół (dla potomka i rodzica)
- Jeśli kolejka zadań aktywnych pusta – zmiana epoki (tablica priorytetów zadań przeterminowanych staje się tablicą zadań aktywnych) – wymiana wskaźników

DYNAMICZNY PRIORYTET PROCESÓW ZWYKŁYCH

- Początkowy priorytet =nice
- Zmiany priorytetu:
  - pri+=5 / dla zadań ograniczonych przez procesor
  - pri=-=5 / dla zadań ograniczonych przez we/wy (interaktywne – ale multimedia, bazy danych?)
  - bez zmian
- Kwant czasu ~ 1/pri (10ms – 200ms)
- Miara aktywności procesu – atrybut sleep_avg <0; 10ms> zmniejszany w czasie wykonywania, zwiększany po obudzeniu

PROCESY RT

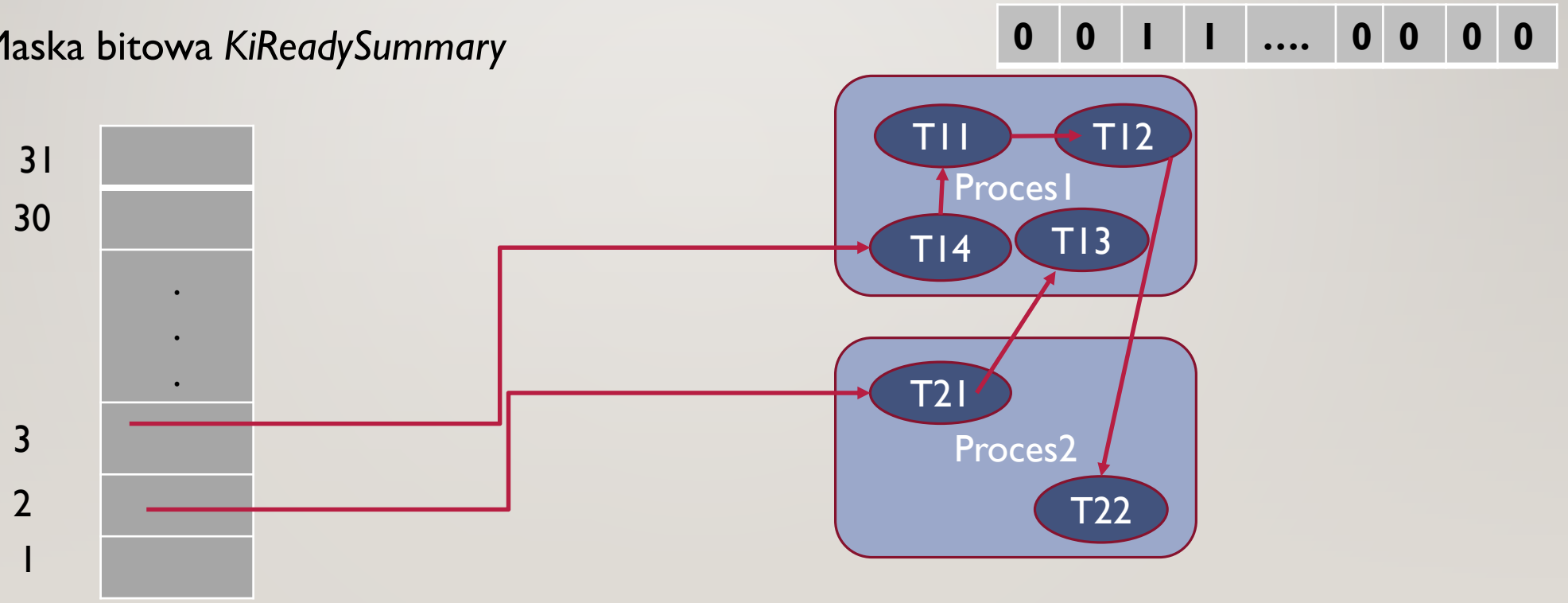
1. SHED_FIFO --- bez wywłaszczania
2. SHED_RR – z wywłaszczaniem po upływie kwantu czasu

WINDOWS 2000

- Szeregowanie na poziomie wątków
- 32 poziomy priorytetów (2 pasma)
  - 0 – bezczynność
  - 1 – 15 – dynamiczne --- algorytm rotacyjny; priorytety dynamiczne
  - 16 – 31 – RT --- algorytm rotacyjny, priorytety statyczne
- Wyższy poziom – wyższy priorytet
- priorytet = klasa + modyfikator dla wątku
- Klasy: idle (4), below normal (6), normal (8), above normal (10), high (13), realtime (24)
- Modyfikator: <-15; 15>

STRUKTURY DANYCH – GOTOWE WĄTKI

- Tablica kolejek wątków gotowych *KiDispatcherReadyListHead*
- Maska bitowa *KiReadySummary*



ZARZĄDZANIE WĄTKAMI

- Przełączenie kontekstu
  - Zakończenie wątku
  - Przejście w stan oczekiwania
  - Wywłaszczenie
  - Upłyniecie kwantu czasu (niekoniecznie przełączenie kontekstu; wątek może być kontynuowany)
- Wywłaszczony wątek trafia na początek kolejki
- Szeregowanie: dwie funkcje:
  - FindREadyThred – po zwolnieniu procesora przez wątek: szuka wątku gotowego o najwyższym priorytecie
  - ReadyThread – dla wątku po przejściu w stan gotowości lub zmianie priorytetu – albo ustawia go do kolejki albo wywłaszcza bieżący wątek, jeśli ma niższy priorytet

WYWŁASZCZANIE PRZEZ WĄTEK

Tg – wątek gotowy

Tw – wątek wykonywany

pri(T) – priorytet wątku T

```
if ( pri(Tg) > pri(Tw))
        if (liczba wykorzystanych kwantów czasu przez Tw >=1)
            umieść Tw na końcu KiDispacherReadyListHead[pri(Tw)]
        else
            umieść Tw na początku KiDispacherReadyListHead[pri(Tw)]
else
        umieść Tg na końcu KiDispacherReadyListHead[pri(Tg)]
```

KWANT CZASU PROCESORA

- Kwant czasu – 6 jednostek (PC); 36 jednostek (serwer)
- 1 takt zegara ---- (-3 jednostki)
- Wyzerowanie czasu – wywłaszczenie (inny wątek ma taki sam priorytet) lub nie; może być obniżony priorytet
- if (pri(T) < 16 && T wchodzi w stan oczekiwania) kwant czasu zmniejszony o 1
- if (pri(T) >14 && T wchodzi w stan oczekiwania || wątek pierwszoplanowy (w pewnych przypadkach) kwant czasu jest odnawiany przed -1
- if (pri(T) < 15 && T wychodzi ze stanu oczekiwania) kwant czasu jest odnawiany
- Wątki pierwszoplanowe mogą uzyskać 3xdłuższy kwant czasu (ustawienia rejestru)
- Wątki długo oczekujące (ponad 300 taktów) – mają 2x wydłużony czas i pri=15 (na jeden kwant)

ZMIANA PRIORYTETÓW WĄTKÓW

- Tymczasowe podwyższenie priorytetu (max. do 15)
  - Po zakończeniu we/wy (o 1-8) / faworyzowanie procesów ograniczonych we/wy i interakcyjnych
  - Po oczekiwaniu na semafor lub zdarzenie (o1, tylko na 1 kwant czasu), kwant czasu zmniejszany o 1
  - Po zakończeniu oczekiwania przez wątek 1-planowy (w zależności od zmiennej w rejestrze; możliwe zwiększenie kwantu czasu dla wszystkich wątków procesu 1-planowego))
  - Po przebudzeniu wątku GUI (o 2)
  - Po długim oczekiwaniu w stanie gotowości (15 na 1 kwant czasu)
- Priorytet sukcesywnie obniżany o 1 po upływie kwantu czasu aż do wartości bazowej