

Systemy operacyjne Wykład 04

Wersja 2024

dr inż. Marek Wilkus <http://home.agh.edu.pl/~mwilkus>
Wydział Inżynierii Metali i Informatyki Przemysłowej
AGH Kraków

Na podstawie programu opracowanego przez dr inż. Krzysztofa Wilka

1

Planowanie przydziału procesora

2

Planowanie przydziału procesora

- W pamięci operacyjnej znajduje się kilka procesów jednocześnie.
- Kiedy jakiś proces musi czekać, system operacyjny odbiera mu zasoby procesora i przekazuje do dyspozycji innego procesu.
- Planowanie przydziału procesora jest jedną z podstawowych funkcji każdego systemu operacyjnego.

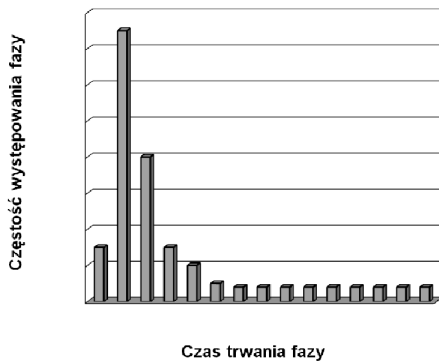
3

Fazy procesora i I/O

załaduj przechowaj dodaj przechowaj czytaj z pliku	Faza procesora
Czekaj na urządzenie wejścia-wyjścia	Faza wejścia/wyjścia
przechowaj dodaj zwiększ indeks pisz do pliku	Faza procesora
Czekaj na urządzenie wejścia-wyjścia	Faza wejścia/wyjścia
załaduj przechowaj dodaj przechowaj czytaj z pliku	Faza procesora
Czekaj na urządzenie wejścia-wyjścia	Faza wejścia/wyjścia

4

Czasy faz procesora



5

Planowanie przydziału procesora

- Proces ograniczony przez I/O ma zazwyczaj dużo krótkich faz procesora.
- Proces ograniczony przez procesor może mieć mało, lecz bardzo długich faz procesora.

6

- Decyzje o zmianie przydziału procesora podejmowane są gdy:
 - Proces przeszedł od stanu aktywności do czekania, np. na zakończenie potomka lub "zamówił" I/O.
 - Proces przeszedł od stanu aktywności do gotowości, np. wskutek wystąpienia przerwania.
 - Proces przeszedł od stanu czekania do gotowości, np. po zakończeniu operacji I/O.
 - Proces kończy działanie.
- Jeśli planowanie odbywa się tylko w pierwszym i ostatnim przypadku, mamy do czynienia z **planowaniem niewyłączeniowym (co-operative)**. W przeciwnym wypadku planowanie jest **wyłączeniowe (preemptive)**.

7

- Bez wyłączzeń - proces, który otrzymał procesor, odda go dopiero przy przejściu do stanu oczekiwania lub po zakończeniu. Nie wymaga zegara. Stosowane w starszych systemach Windows.
- Jednakże planowanie wyłączające jest **ryzykowne**: Należy brać pod uwagę fakt, że proces w momencie wyłączzenia może być w trakcie wykonywania funkcji systemowej.
- Zabezpieczenia:
 - System czeka z przełączeniem kontekstu do zakończenia funkcji,
 - Przerwania są blokowane przy przejściu do ryzykownych fragmentów kodu jądra,
 - Nie wyłącza procesa, gdy wewnętrzne struktury jądra są niespójne.

8

- Jest to moduł, który przekazuje procesor do dyspozycji procesowi wybranemu przez planistę krótkoterminowego. Do jego zadań należy:
 - Przełączanie kontekstu,
 - Przełączanie procesu do trybu użytkownika,
 - Wykonanie skoku w kodzie procesu by wznowić jego działanie.
- Opóźnienie ekspedycji - czas jaki ekspedytor używa na wstrzymanie jednego procesu i wznowienie innego. Czas ten powinien być jak najkrótszy (ekspedytor jak najszybszy).

9

- Wykorzystanie procesora - w realnych systemach od 40 (słabe) do 90% (intensywne),
- Przepustowość - mierzona ilością procesów kończonych w jednostce czasu (dla długich - kilka na godzinę, dla krótkich - kilka na sekundę),
- czas cyklu przetwarzania - od nadejścia procesu do systemu, do jego zakończenia (suma czasów oczekiwania na wejście do pamięci, w kolejce procesów gotowych, okresów aktywności i operacji wejścia-wyjścia),
- Czas oczekiwania - suma czasów w których proces czeka w kolejce p. gotowych,
- Czas odpowiedzi - w systemach interakcyjnych - czas od wysłania żądania do rozpoczęcia odpowiedzi.

10

- **Maksymalne** wykorzystanie procesora,
- **Maksymalna** przepustowość,
- **Minimalny** czas cyklu przetwarzania,
- **Minimalny** czas oczekiwania,
- **Minimalny** czas odpowiedzi.

11

Zgłaszają się 3 procesy:

1. P1 o czasie trwania fazy: 24 ms
2. P2 o czasie trwania fazy: 3 ms
3. P3 o czasie trwania fazy: 3 ms

Diagram Gantta dla planowania FCFS:



Średni czas oczekiwania wynosi więc:

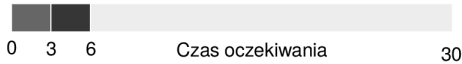
$$(0+24+27)/3 = 17 \text{ ms}$$

12

A gdyby procesy zgłosiły się w kolejności:

1. P2
2. P3
3. P1

Diagram Gantta dla planowania FCFS:



Średni czas oczekiwania wyniósłby wtedy:

$$(0+3+6)/3 = 3 \text{ ms}$$

13

- Pierwszy przykład pokazuje, że planowanie metodą FCFS nie jest optymalne.
- Średni czas oczekiwania bardzo zależy od kolejności zgłoszenia procesów.
- Efekt konwoju - małe procesy czekają na zwolnienie procesora przez jeden wielki proces.
- Algorytm FCFS jest niewyłączający.
- Algorytm ten jest nieużyteczny w systemach z podziałem czasu, w których procesy powinny dostawać procesor w regularnych odstępach czasu.

14

Metoda SJF (Najpierw najkrótsze zadanie)

Zgłaszają się cztery procesy:

1. P1 o czasie trwania fazy: 6 ms
2. P2 o czasie trwania fazy: 8 ms
3. P3 o czasie trwania fazy: 7 ms
4. P4 o czasie trwania fazy: 3 ms

Diagram Gantta dla planowania SJF:



Średni czas oczekiwania wynosi:
 $(3+16+9+0)/4 = 7 \text{ ms}$

Dla metody FCFS wyniósłby:
 $(0+6+14+21)/4 = 10,25 \text{ ms}$

15

Planowanie metodą SJF

- Można udowodnić, że planowanie tą metodą jest optymalne.
- Umieszczenie krótkiego procesu przed długim w większym stopniu zmniejsza czas oczekiwania krótkiego procesu niż zwiększa czas oczekiwania procesu długiego.
- Algorytm SJF jest często używany przy planowaniu długoterminowym.
- Problem - **nie można przewidzieć długości następnej fazy procesora**, można ją tylko szacować, najczęściej za pomocą średniej wykładniczej z poprzednich faz procesora.

16

Planowanie metodą SJF

$$f_{n+1} = \alpha t_n + (1-\alpha) f_n$$

gdzie:

α - współczynnik wagi (0÷1)

t_n - długość n-tej fazy procesora,

f_n - informacje o poprzednich fazach

gdy $\alpha = 0$ - bierzemy pod uwagę tylko dawniejszą historię,

a gdy $\alpha = 1$ - bierzemy pod uwagę tylko ostatnie fazy.

Najczęściej $\alpha = 0,5$

17

Planowanie metodą SJF

- Algorytm SJF może być wyłączający lub niewyłączający.
- Gdy do kolejki dochodzi nowy proces, który posiada fazę procesora krótszą niż pozostały czas w bieżącym procesie, algorytm wyłączający odbiera procesor bieżącemu procesowi i przekazuje go krótszemu.
- Metoda ta nazywa się **SRTF** (shortest remaining time first) czyli „najpierw najkrótszy pozostały czas”.
- Algorytm niewyłączający pozwala na dokończenie fazy procesora.

18

- Mechanizm bardzo podobny do SJF, ale kryterium szeregowania jest priorytet procesu.
- Najpierw wykonywane są procesy o ważniejszym priorytecie.
- Priorytety mogą być definiowane wewnętrznie, na podstawie pewnych cech procesu (np. wielkość pamięci, limity czasu, zapotrzebowanie na urządzenia we-wy itd..)
- Priorytety definiowane zewnętrznie mogą np. zależeć od ważności użytkownika, jego firmy czy też od innych politycznych uwarunkowań.

19

- Wady:
 - Procesy o niskim (mało ważnym) priorytecie mogą nigdy nie dostać procesora (głodzenie, nieskończone blokowanie).
 - Przykład: w 1973 r. w wycofywanym z eksploatacji na MIT komputerze IBM 7094 wykryto „zagłodzony” proces o niskim priorytecie, który został uruchomiony do wykonania w 1967 roku ;-(.
- Rozwiązaniem problemu jest „postarzanie” czyli podnoszenie priorytetu procesów oczekujących zbyt długo.
- Przykład: przydział mieszkania dla dziadka w "Alternatywy 4".

20

- Zaprojektowane specjalnie do systemów z podziałem czasu.
- Każdy proces otrzymuje kwant czasu (w starszych rozwiązaniach 10-100ms), po upływie którego jest wyłączeni i umieszczany na końcu kolejki zadań gotowych.
- Kolejny proces do wykonania jest wybierany zgodnie z algorytmem FCFS.
- Jeżeli jest n procesów gotowych a kwant czasu wynosi q , to każdy proces czeka nie dłużej niż $(n-1)*q$ jednostek czasu.

21

- | | |
|-------|------------------------------|
| 1. P1 | o czasie trwania fazy: 24 ms |
| 2. P2 | o czasie trwania fazy: 3 ms |
| 3. P3 | o czasie trwania fazy: 3 ms |

Diagram Gantta dla $q=25$ ms:

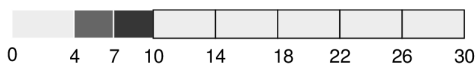


Średni czas oczekiwania wynosi:
 $(0+24+27)/3 = 17$ ms
 Przetęczań kontekstu: 2

22

- | | |
|-------|------------------------------|
| 1. P1 | o czasie trwania fazy: 24 ms |
| 2. P2 | o czasie trwania fazy: 3 ms |
| 3. P3 | o czasie trwania fazy: 3 ms |

Diagram Gantta dla $q=4$ ms:

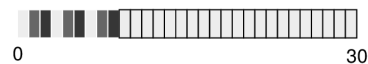


Średni czas oczekiwania wynosi:
 $(17)/3 = 5,66$ ms
 Przetęczeń kontekstu: 7

23

- | | |
|-------|------------------------------|
| 1. P1 | o czasie trwania fazy: 24 ms |
| 2. P2 | o czasie trwania fazy: 3 ms |
| 3. P3 | o czasie trwania fazy: 3 ms |

Diagram Gantta dla $q=1$ ms:



Przetęczeń kontekstu: 29

24

Planowanie rotacyjne

- Wydajność algorytmu zależy bardzo od kwantu czasu q .
 - Gdy q jest duże, algorytm RR przechodzi praktycznie w algorytm FCFS.
 - Gdy q jest bardzo małe, to przez znaczną część czasu procesor zajęty jest przełączaniem kontekstu.
- Ogólna zasada:
 - 80% faz procesora powinno być krótszych niż kwant czasu.

25

Wielopoziomowe planowanie kolejek

- Kolejka procesów gotowych jest podzielona na oddzielne pod-kolejki, na przykład:
 - Kolejka procesów pierwszoplanowych (foreground),
 - Kolejka procesów drugoplanowych (background).
- Przykładowe proponowane algorytmy planowania:
 - Procesy pierwszoplanowe: RR
 - Procesy drugoplanowe: FCFS
- Procesy pierwszoplanowe mają absolutny priorytet nad drugoplanowymi.
- Aby nie „zagłodzić” procesów 2-planowych, stosuje się podział czasu na kolejki, przykładowo:
 - Kolejka pierwszoplanowa - 80% czasu procesora,
 - Kolejka drugoplanowa - pozostałe 20%

26

Kolejki wielopoziomowe ze sprzężeniem zwrotnym

- Mechanizm ten pozwala na przesuwanie procesów pomiędzy kolejkami.
- Proces, który używa za dużo procesora, można „karnie” przenieść do kolejki o niższym priorytecie i przez to dać szerszy dostęp do procesora innym procesom.
- Dzięki temu procesy ograniczone przez we-wy i procesy interakcyjne mogą pozostać w kolejkach o wyższych priorytetach.
- Długo oczekujące procesy z kolejki niskopriorytetowej mogą być przeniesione do ważniejszej - działa mechanizm postarzania procesów (przeciwdziała ich głodzeniu).
- **Planowanie ze sprzężeniem zwrotnym jest najbardziej złożonym algorytmem planowania przydziału procesora.**

27

Kolejki wielopoziomowe ze sprzężeniem zwrotnym - przykład

- Kolejka trzydziomowa: K0, K1, K2
- Proces wchodzący trafia do kolejki K0 i dostaje kwant czasu 8 ms.
- Jeśli nie zakończy się w tym czasie, jest wyrzucany na koniec niższej kolejki K1.
- Gdy kolejka K0 się opróżni i przyjdzie czas wykonywania naszego procesu, dostaje on kwant czasu 16 ms.
- Jeśli i w tym czasie proces nie skończy działania, jest wyrzucany na koniec kolejki K2, obsługiwanej w porządku FCFS (oczywiście pod warunkiem, że kolejki K0 i K1 są puste).
- Tak więc najszybciej wykonywane są procesy do 8 ms, nieco wolniej procesy od 8 do $8+16=24$ ms, a najdłużej czekają procesy długie (są obsługiwane w cyklach procesora nie wykorzystanych przez kolejki 1 i 2).

28

Planowanie wieloprocessorowe

- Planowanie **heterogeniczne** – dla systemów sieciowych, rozproszonych, o różnych procesorach – trudne w realizacji.
- Planowanie **homogeniczne** – dla procesorów tego samego typu – nie stosuje się różnych kolejek per procesor ze względu na możliwość przestoju procesorów.
- SMP – wieloprzetwarzanie symetryczne – każdy procesor wybiera sam proces ze wspólnej kolejki – działanie musi być tak zaprogramowane, by uniknąć np. dublowania.
- Wieloprzetwarzanie asymetryczne – Jeden procesor nadrzędny przydziela zadania innym procesorom (hierarchia aktywny/pasywny, master/slave). Pozostałe wykonują zadania.

29

Planowanie w czasie rzeczywistym

- W rygorystycznych systemach czasu rzeczywistego zadanie krytyczne musi zostać wykonane w odpowiednim momencie i terminie. Zasoby dla tego zadania są więc rezerwowane. W przypadku niemożności zarezerwowania zasobów, planista rezygnuje z przydziału zadania do procesora.
- W łagodnych systemach czasu rzeczywistego procesy RT mają wyższy priorytet niż pozostałe. Te priorytety nie mogą ulec obniżeniu. W krytycznych sytuacjach musimy zezwolić na wyłączenie działających funkcji systemowych (zapewnić odpowiednie punkty wyłączenia/przywracania) lub jądra systemu (niezbędne mechanizmy synchronizacji).
- Wysokopriorytetowe procesy **nie mogą czekać** na zakończenie niskopriorytetowych.

30

- **Modelowanie deterministyczne** – zakładamy z góry konkretne obciążenie systemu i definiujemy zachowanie się poszczególnych algorytmów w tych warunkach (np. wyznaczanie średniego obciążenia z diagramu Gantta).
- **Modele obsługi kolejek** – na podstawie analizy obsługi kolejek w sieciach:
 Wzór Little'a: $n=L*W$
 (n – średnia długość kolejki, L – liczba nowych procesów/s, W – średni czas oczekiwania w kolejce)
- **Symulacje** na zaprogramowanym modelu systemu i danych symulacyjnych. Dane otrzymywane są np. z zapisów działania rzeczywistych systemów.
- **Implementacja** – algorytm implementuje się w rzeczywistym systemie i przeprowadza stosowne pomiary.

31

- Planowanie w wielopoziomowych kolejkach ze sprzężeniem zwrotnym,
- 4 klasy procesów: **real time, system, time sharing, interactive,**
- Priorytet globalny i priorytety w obrębie klas,
- Proces potomny dziedziczy klasę i priorytet,
- Klasa domyślna - **time sharing,**
- Im większy priorytet, tym mniejszy kwant czasu
- Klasa **system** - procesy jądra,
- Klasa **interactive** - wyższy priorytet mają aplikacje graficzne X11,
- Wątki o tym samym priorytecie planowane są algorytmem RR.

32

- CFS** - Completely Fair Scheduler - wykorzystuje jako kolejkę procesów gotowych **drzewiastą strukturę** (drzewa czerwono-czarne) szybkie przy wyszukiwaniu, ale $O(\log n)$ przy zapisie.
- Najpierw przydział dostają procesy o najniższym otrzymanym dotychczasowo czasie wirtualnym (vruntime), przechowywanym w nanosekundach.
 - Jeżeli proces zakończył działanie, usuwany jest zupełnie z kolejki.
 - Jeżeli proces osiągnął swój maksymalny czas działania lub został wstrzymany / oczekuje, jest umieszczany w drzewie-kolejce w miejsce określone za pomocą czasu przydziału zaktualizowanego o bieżąco wykorzystany. Następnie wybierany jest kolejny proces o najmniejszym czasie przydziału (co w drzewie ma niewielki koszt obliczeniowy).

33

- vruntime określany jest z czasu działania danego procesu w stosunku do liczby aktualnie działających procesów o tym samym współczynniku redukcji.
- **Priorytety** - są współczynnikami redukującymi otrzymany czas (niższy priorytet - bardziej zredukowany czas).
- Konfiguracja - dostosowanie dwóch opcji:
 - Latency - maksymalne dopuszczalne opóźnienie (mniejsze dla środowisk interakcyjnych, większe w węzłach obliczeniowych, serwerach)
 - Granularity - minimalny czas działania procesu, by system nie zajmował się tylko przełączaniem.
- Przy czym procesy wstrzymujące działanie na mniej niż swój kwant czasu mają czas proporcjonalny do tej wartości odejmowany od vruntime (nie blokować ciągłym przełączaniem kontekstu!).

34

- Każdy proces ma priorytet *statyczny* od 100 (najwyższy) do 139 (najniższy).
- Podstawowy kwant wyliczany jest ze wzoru:

$$\text{base time quantum (in milliseconds)} = \begin{cases} (140 - \text{static priority}) \times 20 & \text{if } \text{static priority} < 120 \\ (140 - \text{static priority}) \times 5 & \text{if } \text{static priority} \geq 120 \end{cases}$$
- Czyli wyższy priorytet → większy kwant.
- Każdy proces ma również priorytet *dynamiczny* używany przy wyborze z kolejki procesów gotowych:

$$\text{dynamic priority} = \max(100, \min(\text{static priority} - \text{bonus} + 5, 139))$$

Bonus - 0..10, <5 zmniejsza priorytet, >5 podnosi go. Zależy od średniego czasu wstrzymywania procesu ze względu na I/O.

35

Average sleep time	Bonus	Granularity
Greater than or equal to 0 but smaller than 100 ms	0	5120
Greater than or equal to 100 ms but smaller than 200 ms	1	2560
Greater than or equal to 200 ms but smaller than 300 ms	2	1280
Greater than or equal to 300 ms but smaller than 400 ms	3	640
Greater than or equal to 400 ms but smaller than 500 ms	4	320
Greater than or equal to 500 ms but smaller than 600 ms	5	160
Greater than or equal to 600 ms but smaller than 700 ms	6	80
Greater than or equal to 700 ms but smaller than 800 ms	7	40
Greater than or equal to 800 ms but smaller than 900 ms	8	20
Greater than or equal to 900 ms but smaller than 1000 ms	9	10
1 second	10	10

36

CFS - proces interaktywny

- Proces zostaje uznany za „interaktywny” gdy spełnione jest:

$$\text{dynamic priority} \leq 3 \times \text{static priority} / 4 + 28$$

- Czyli:

$$\text{bonus} - 5 \geq \text{static priority} / 4 - 28$$

interactive
delta

- Bonus < 5 powoduje, że proces nie „zajmie” czasu innym.
- Bonus ≥ 5 jest dla procesów o wyższym czasie aktywnego oczekiwania.
- Procesy interaktywne są umieszczane zawsze w aktualnie planowanej kolejce.

37

Przełączenie kontekstu w CFS

- Aktualizujemy licznik kwantu czasu.
- Jeżeli proces wyczerpał swój czas, to musimy przełączyć proces:
- Usuń proces z listy procesów aktywnych.
- Zaznacz proces do ponownego przeliczenia czasu.
- Zaktualizuj priorytet dynamiczny procesu.
- Zresetuj licznik kwantu czasu.
- Oznacz proces jako zakończony lub przełączony.
- Wstaw wywłaszczony proces do listy aktywnych lub zakończonych.
- PRZEŁĄCZ PROCESY (zapis bloku kontrolnego, przywołanie bloku, wznowienie).

Linux - specjalne procesy

- Możliwe jest zdowngrade’owanie metody planowania procesu. W ten sposób można naiwnie utworzyć klasy procesów.
- SCHED_FIFO - kolejka procesów planowanych metodą FIFO może uruchamiać się gdy kolejka innego planisty nie wymaga tego.
- SCHED_RR - Kolejka procesów planowanych RR.

39

Planowanie w systemach Windows

- Priorytetowe z wywłaszczeniami,
- NT: 32 kolejki procesów, 6 klas priorytetów, 7 względnych priorytetów w obrębie klas,
- Priorytet domyślny w klasie: **normal**.
- Wątek jest wykonywany aż:
 - Zostanie wywłaszczony przez proces o wyższym priorytecie,
 - Zużyje swój kwant czasu,
 - Wykonuje operacje I/O blokujące zasoby lub wykorzystuje taki sterownik (tryby zgodności),
 - Zakończy się.
- Proces powiązany z aktywnym oknem ma zwiększany kwant czasu (NT5.1 – trzykrotnie).

40

Windows NT i późniejsze

- Rotacyjnie (Round-robin) z dodatkowymi priorytetami i wielopoziomą kolejką z informacją zwrotną.
- Niektóre procesy mają intencjonalnie zmniejszone priorytety (procesy z harmonogramu zadań, defragmentator dysków). Kryterium jest proces nadrzędny lub charakterystyka programu.
- Windows 10: Wielopoziomowe kolejki ze sprzężeniem zwrotnym.

41

Planowanie w Systemie HaikuOS

- 3 możliwe strategie: simple, simple-SMT (dla procesorów wielordzeniowych), affine (wersja bieżąca).
- Wersja docelowa, po wersji beta, aktualnie w implementacji:
 - Dwie klasy procesów: Priorytet (p) 1-99 dostają $k \cdot 2^p$ kwant czasu, powyżej 99 są realtime i inne procesy mogą działać tylko, gdy te zwolnią CPU.
 - Proces może stać się „realtime” w specjalnych okolicznościach związanych z obsługą układów akceleratora i grafiki.
 - Aktywne okno ma większe szanse stać się „realtime”.
 - Serwery i kity nie będące częścią jądra mogą stać się „realtime” używając funkcji jądra.

42

- Proces na pierwszym planie ma pierwszeństwo.
- Procesy odsyłane do tła mogą mieć czas "karny", przez który nie dostaną procesora.
- Procesy w tle mogą mieć ograniczony dostęp do API systemowego (np. tylko powiadomienia i sieć).

- Przyjmując roboczo, że proces w pamięci stałej urządzenia mobilnego to niemal to samo co proces odswap-owany na dysk (niewielkie koszty czasowe załadowania procesu z SSD), w wielu systemach mobilnych **planista średnioterminowy** dodaje w miarę potrzeb ważniejsze usługi systemowe.
- Podejście to gwarantuje, że usługa systemowa potrzebująca akurat przetwarzać nadchodzące dane (łączność, I/O, telemetria) uruchomi się nawet, gdy jej proces nie działa.
- Stosowane jedynie dla nielicznych, krytycznych usług systemu.

- Wiele programów działa w tle, a ich pełne zakończenie występuje rzadko – stąd planista powinien uwzględniać taką sytuację.
- Procesy te, gdy działają w tle, nie wymagają zupełnie planowania jak procesy interakcyjne – stąd Symbian przeniósł je „z automatu” do kolejek o niższych priorytetach.
- Android/iOS dostosowują położenie tych procesów dynamicznie.

Dziękuję za uwagę