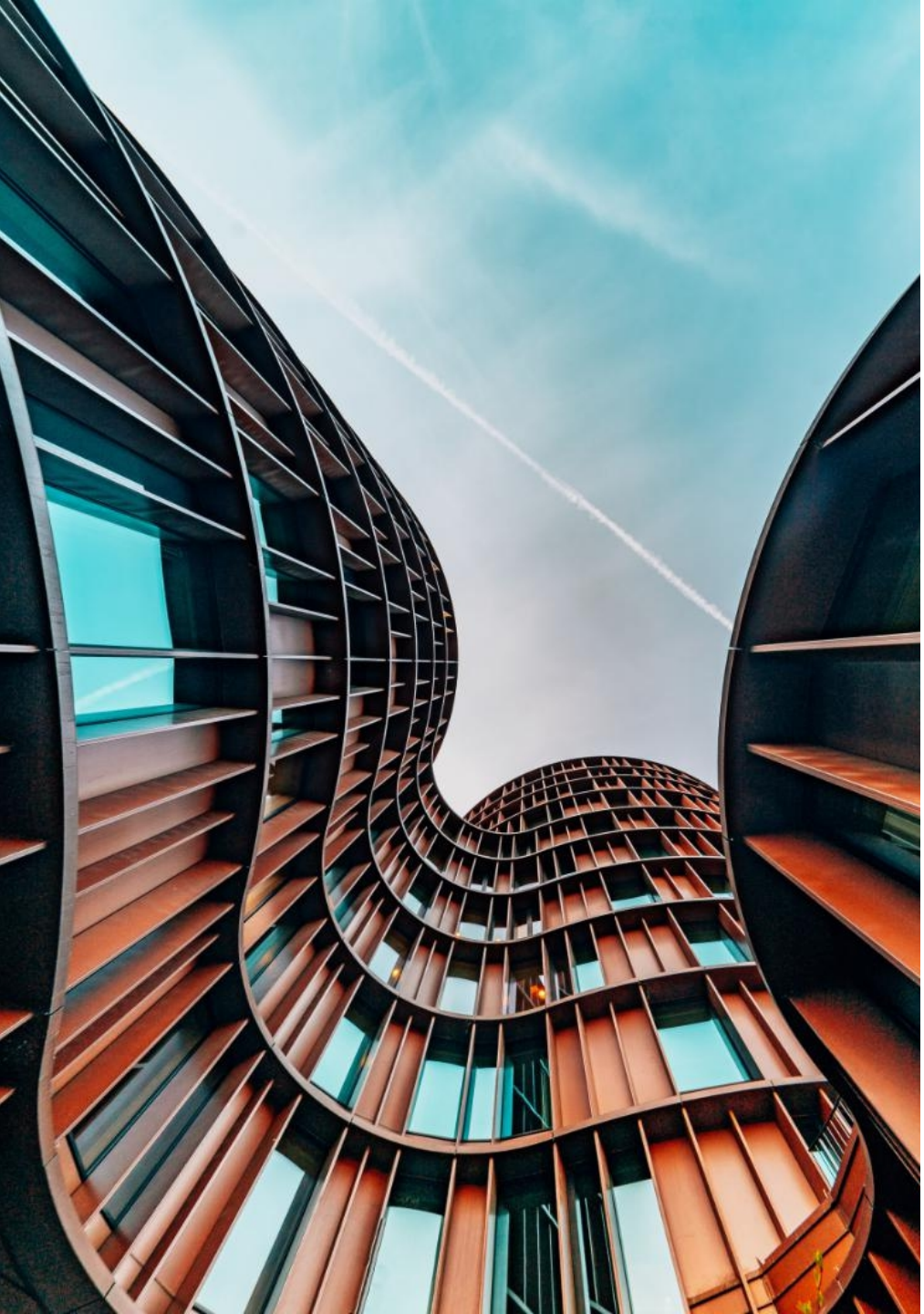


Тема 4

Планування та диспетчеризація

План

1. Основні поняття планування.
2. Особливості планування у різних типах ОС.
3. Приклади алгоритмів планування.

1. Основні поняття планування

? Використання яких саме системних ресурсів в ОС *планується*?

Передусім тих, керування якими відбувається в часі: ЦП, пристрої введення-виведення.

У цій темі розглядаємо ***планування використання ЦП***.

? Планування чого - процесів чи потоків?

Можливі обидва варіанти.

Основні тези:

- якщо ОС підтримує потоки на рівні ядра - вона має вміти їх планувати;
- там, де не принципово - процеси чи потоки, - говоритимемо про процеси;
- там, де принципово, будемо говорити про потоки.

1. Основні поняття планування

З урахуванням усього сказаного раніше:

Планування - встановлення послідовності виконання процесів на ЦП.

Планувальник і диспетчер

Планувальник (*scheduler*) вибирає процес, що має виконуватися наступним.

Планувальник діє за **алгоритмами планування** (часто за кількома).

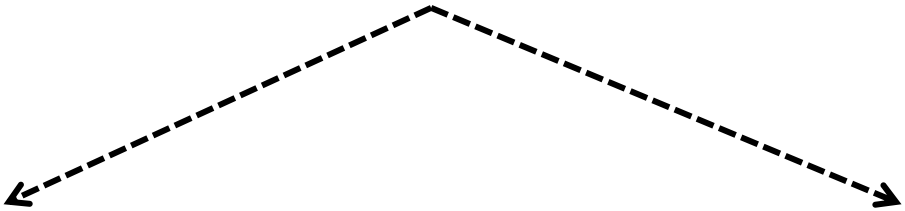
Диспетчер (*dispatcher*) дає змогу процесу, вибраному планувальником, почати виконуватися на ЦП шляхом перемикання контексту.

Що саме робить диспетчер:

- перемикає контекст з одного процесу на інший;
- перемикається у режим користувача;
- переходить на те місце у програмному коді, звідки треба почати (продовжити) виконання процесу.

1. Основні поняття планування

Перемикання контексту



Добровільне

Процес перейшов у стан очікування (сам) або самостійно завершився.

Примусове

Процес перервано системою (інший, важливіший процес; вичерпано відведений квант часу тощо).

1. Основні поняття планування

Планування без витіснення (*non-preemptive scheduling*)

Процесу надається можливість виконуватися:

- поки процес не завершиться сам;
- поки процес не перейде у стан очікування.

Має місце ***добровільне перемикання контексту***.

Планування з витісненням (*preemptive scheduling*)

Виконання процесу може бути перервано. Право на виконання передається іншому процесу.

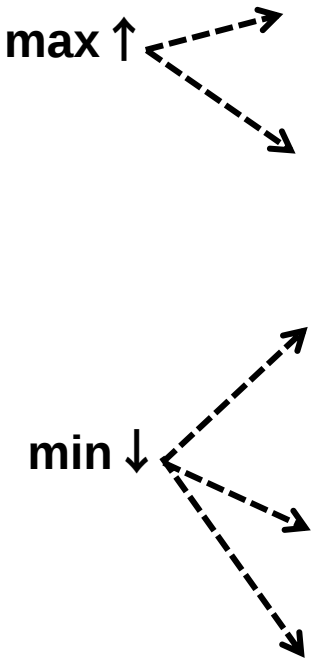
Не лише добровільне, а й ***примусове перемикання контексту***.

Основна вимога до диспетчера - працювати дуже швидко (бо перемикання контексту може відбуватися часто).

1. Основні поняття планування

Вимоги до планувальника менш однозначні.

Основні критерії планування



- **Використання ЦП** (%)

  Який відсоток часу ЦП зайнятий (Приблизно у межах 40-90%)

- **Пропускна здатність** (*throughput*)

  Кількість процесів, завершених за одиницю часу.

- **Час відгуку** (*response time*)

  Час від надходження запиту до початку відповіді.

  Показує, як швидко реагує система.

- **Час очікування** (*waiting time*)

  Час, який процес проводить у чергах (весь час, в усіх чергах).

- **Час обороту** (*turnaround time*)

  Час, витрачений на виконання процесу (виконання на ЦП, очікування на введення-виведення, очікування в чергах).

1. Основні поняття планування

Підсумуємо вимоги:

- ЦП не має простоювати, якщо є невиконані завдання;
- система має бути готова оперативно реагувати на нові запити;
- виконати якомога більше процесів в одиницю часу;
- виконувати процес швидко;
- не тримати процес у черзі надто довго.

+ ОС можуть мати додаткові вимоги до планування.

Наприклад:

в інтерактивних системах -
краще зменшувати не максимальний чи середній час відгуку, а його дисперсію (наскільки окремі значення часу очікування відрізняються від середнього);

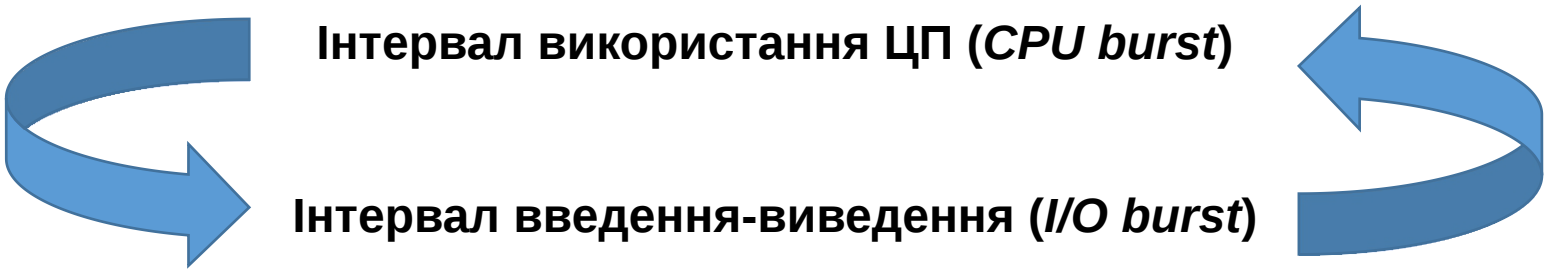
у системах реального часу -
процеси мають виконуватися у визначені строки.

+ Керуватися здоровим глуздом:

Затрати ресурсів на виконання процесів з плануванням не мають перевищувати затрати на виконання процесів без планування.

1. Основні поняття планування

Виконання процесу є чергуванням двох інтервалів:



Залежно від виду активності, що переважає, виокремлюють:

- **процеси, обмежені швидкістю обчислень**
  (переважає використання ЦП);
- **процеси, обмежені швидкістю введення-виведення**
  (переважає використання пристроїв введення-виведення).

1. Основні поняття планування

(1) Процес, обмежений швидкістю обчислень (*CPU bound process*):

(2) Процес, обмежений швидкістю введення-виведення (*I/O bound process*):

- Що швидше працює ЦП - то більше процесів категорії (1) переходять у категорію (2).
- Пристрої введення-виведення працюють повільніше за ЦП.
- Розподіл періодів обчислень / очікування на введення-виведення може впливати на вибір алгоритмів планування.

Схеми адаптовано за джерелом:
Tanenbaum, Bos. Modern Operating Systems, 2014

1. Основні поняття планування

Щоб краще розібратися з наступним питанням, розглянемо ще два типових стани процесів:

- **Очікування / Відкладено** (*Blocked / Suspended*)
- **Готовий до виконання / Відкладено** (*Ready / Suspended*)

Навіщо:

процеси, які зараз не виконуються, можна тимчасово вивантажувати на диск - це **підкачування** (*swapping*)

зменшення використання оперативної пам'яті

Потім процес повертається в оперативну пам'ять.

Підкачування пов'язано з:

- керуванням пам'яттю (див. *тема 5*);
- роботою планувальника (окремі черги для відкладених процесів).

1. Основні поняття планування

Види планування (за тривалістю)

Довготермінове планування (*long-term scheduling*)

Вибирає процес з черги завдань (job queue) і поміщає його у чергу процесів, готових до виконання (ready queue).

Вирішує, чи додавати ще один процес до вже наявних? (можуть бути обмеження)

Середньотермінове планування (*medium-term scheduling*)

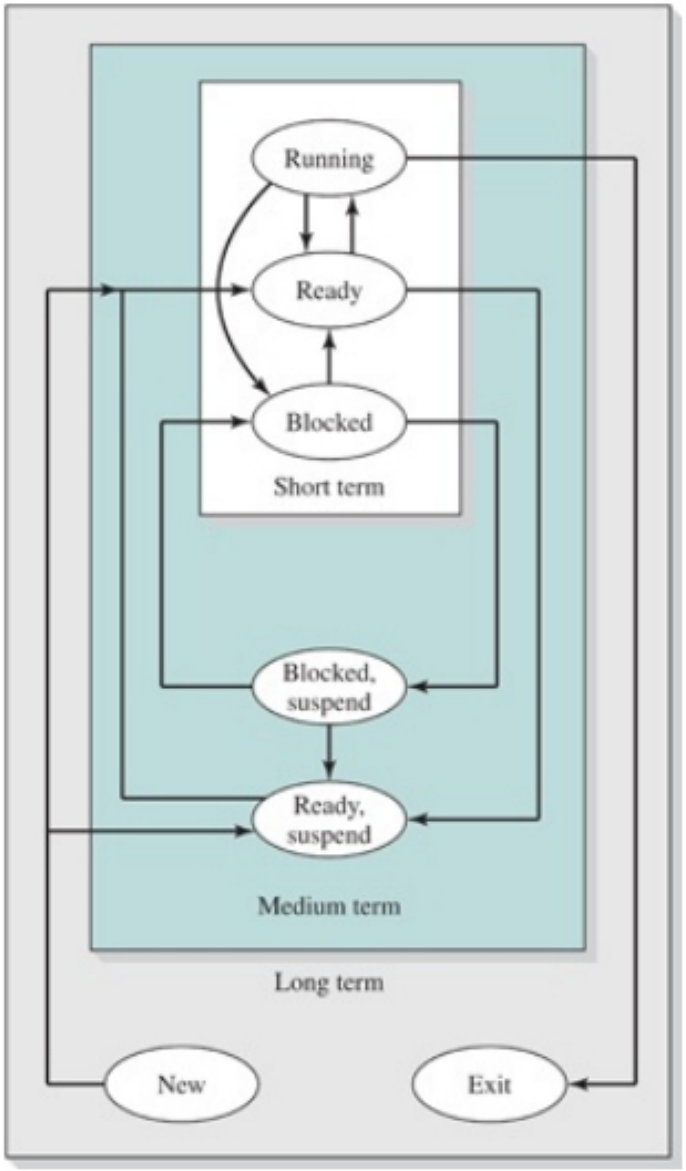
Керує чергами відкладених завдань.

Короткотривале планування (*short-term scheduling*)

Вибирає процес з черги процесів, готових до виконання (ready queue).

Саме цьому процесу передає керування диспетчер.

1. Основні поняття планування



Короткотермінове планування -
основні перемикання між станами:

- Виконання (*Running*);
- Готовий до виконання (*Ready*);
- Очікування (*Blocked*)

Середньотермінове планування -
основні перемикання між станами:

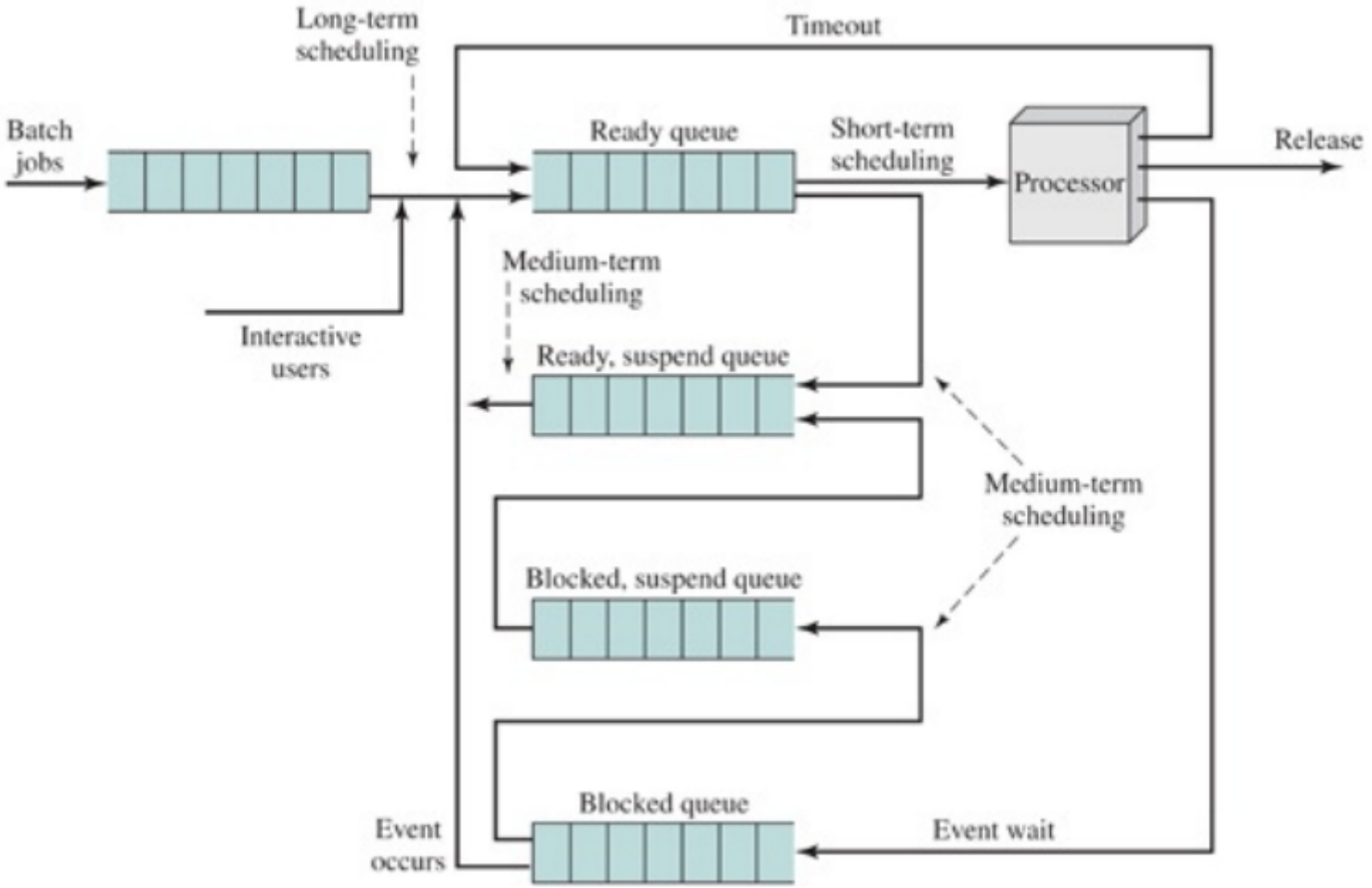
- Готовий до виконання / Відкладено (*Ready / Suspend*);
- Готовий до виконання (*Ready*);
- Очікування / Відкладено (*Bloked / Suspend*);
- Очікування (*Blocked*)

Довготермінове планування -
основні перемикання між станами:

- Створення (*New*);
- Готовий до виконання / Відкладено (*Ready / Suspend*);
- Готовий до виконання (*Ready*);
- Завершення (*Termitated*)

Джерело схеми: W. Stallings, Operating Systems Internals and Design Principles, 9th ed., Pearson, 2017, p. 844, fig. 9.2

1. Основні поняття планування



Джерело схеми: W. Stallings, Operating Systems Internals and Design Principles, 9th ed., Pearson, 2017, p. 845, fig. 9.3

1. Основні поняття планування

Ситуації, в яких вступає в дію планувальник

1. Процес, що виконувався, породив дочірній процес
   (треба вибрати, кому виконуватися у першу чергу - батьківському чи дочірньому)
2. Процес, що виконувався, завершив роботу
   (треба вибрати на його місце інший процес)
3. Процес, що виконувався, перейшов у стан очікування
   (треба вибрати на його місце інший процес)
4. Сталася подія, на яку очікував процес у стані очікування
   (коли йому надати право виконуватися - зараз? пізніше? наскільки пізніше?)
5. Настало чергове переривання по таймеру
   (поточний процес може виконуватися далі? чи його треба перервати?)

2. Особливості планування різних типах ОС

Класифікація ОС
(залежно від характеру отримання ними завдань)

1. ***Системи пакетної обробки.***

   Завдання надходять пакетами. Їх треба виконати якомога швидше. Активної взаємодії з користувачем немає.

2. ***Інтерактивні системи.***

   Завдання надходять будь-коли. Активна взаємодія з користувачем, ~~який не має почуватися покинутим~~
   якому треба забезпечити позитивний користувацький досвід.

3. ***Системи реального часу.***

   Завдання мають виконуватися у визначені строки.

2. Особливості планування різних типах ОС

Планування у системах пакетної обробки

орієнтовані на виконання тривалих задач,
НЕ орієнтовані на користувача

алгоритми без витіснення

або

алгоритми з витісненням з якнайдовшим періодом безперервного виконання

2. Особливості планування різних типах ОС

Планування в інтерактивних системах

орієнтовані на взаємодію з користувачем

алгоритми з витісненням

або

алгоритми без витіснення, але з перериванням тривалого виконання процесів

2. Особливості планування різних типах ОС

Планування у системах реального часу

Витіснення не відіграє суттєвої ролі, оскільки:

- робота процесів спрямована на досягнення спільної мети;
- процеси поводяться передбачувано;
- процеси запускаються ненадовго, виконують свою частину роботи і переходять у стан очікування;
- відсутні сторонні програми.

Види алгоритмів:

- *статичні* (планування – до запуску системи);
- *динамічні* (планування – у ході роботи системи).

Важливо: забезпечити коректну реакцію на зовнішні події (завадити роботі основних процесів!).

не

3. Приклади алгоритмів планування

Алгоритм FIFO (First In - First Out - першим прийшов - першим вийшов)

Інша назва - FCFS (First Came - First Served - першим прийшов - першим обслужили).

Основні особливості

- Черга процесів, готових до виконання, організована за принципом FIFO (звичайна черга).
- Алгоритм без витіснення.
- Новоприбулі процеси поміщаються у хвіст черги.
- Якщо процес А переходить зі стану виконання у стан очікування, право на виконання надається процесу В (голові черги).
- Коли процес А перейде у стан готовності, його буде поміщено у хвіст черги.
- Переваги: простий у реалізації.
- Недоліки: “ефект конвою” (процеси, орієнтовані на введення-виведення, будуть дискримінуватися процесами, орієнтованими на обчислення); алгоритм без витіснення (не підходить для інтерактивних ОС).

3. Приклади алгоритмів планування

Алгоритм SJF (Shortest Job First - спершу найкоротше завдання)

Насправді не найкоротше завдання, а завдання з найкоротшим наступним періодом обчислювальної активності (CPU burst).

Основні особливості

- Серед готових до виконання процесів вибирається той, у якого орієнтовний розмір наступного періоду обчислювальної активності найменший.
- Існує варіант з витісненням і без витіснення.
- Варіант з витісненням має назву SRT (Shortest Remaining Time - найменший час до кінця виконання).
- Якщо з витісненням, то виконуваний процес може витіснити новоприбулий процес з меншим орієнтовним розміром наступного періоду обчислювальної активності.
- Переваги: дозволяє оперативно реагувати на короткі швидкі запити.
- Недоліки: орієнтовний розмір наступного періоду обчислювальної активності складно оцінити (наближені методи).

3. Приклади алгоритмів планування

Алгоритм RR (Round Robin - циклічне планування)

Спирається на припущення, що всі процеси рівноважливі.

Основні особливості

- Кожному процесу виділяється для виконання певний період часу - **квант часу** (*time slice*, *time quantum*)
- Процес може завершитися сам, до того, як буде вичерпано квант.
- Або процес може бути витіснено наступним у черзі процесом, якщо квант вичерпано, а процес ще не завершився.
- Черга - за алгоритмом FIFO, але циклічна.
- Розміри кванту - 10-100 мілісекунд.
- Надто короткий квант - надто часті перемикання контексту.
- Надто довгий квант - деградує у бік звичайного FIFO.
- Гарна практика: 80% періодів обчислювальної активності процесів мають бути коротші за квант.
- Часто використовується у поєднанні з іншими алгоритмами (наприклад, черга не за FIFO)

3. Приклади алгоритмів планування

Алгоритми пріоритетного планування (Priority Scheduling)

Спирається на припущення, що одні процеси важливіші за інші.

Основні особливості

- Кожному процесу присвоюється пріоритет. Процеси з вищим пріоритетом мають перевагу під час вибору планувальником.
- У різних ОС - різна система пріоритетів.
- Один процес може мати декілька пріоритетів.
- Можна групувати процеси у класи з різним пріоритетом.
- У межах таких класів можна застосовувати RR.
- Пріоритети мають переглядатися. Зокрема для уникнення ***голодування*** (*starvation*) - ситуації, за якої процеси з низькими пріоритетами не можуть отримати доступ до ЦП.

3. Приклади алгоритмів планування

Гарантоване планування

Основні особливості

- Гарантується, що кожний з n процесів отримає 1/n частину часу ЦП.
- Відстежується, скільки часу ЦП процес уже використав.
- Використаний час ділиться на розмір частки цього процесу (1/n)
- Наприклад:

  0,3 - процес використав близько третини часу ЦП;

  2,7 - процес перевикористав час ЦП на 1,7, його має бути витіснено.
- На місце витісненого - процес з меншим співвідношенням.

3. Приклади алгоритмів планування

Лотерейне планування

Основні особливості

- Між усіма процесами розігруються лотерейні білети.
- Переможцю - доступ до ресурсів.
- Більш пріоритетним процесам - більше лотерейних білетів.

Справедливе планування

Основні особливості

- Частка часу ЦП надається не процесам, а користувачам.
- Частка часу ЦП може бути використана процесами цього користувача.
- Кількість прцоесів користувача не має значення - головне не перебрати частку.

3. Приклади алгоритмів планування

Особливості планування у системах реального часу

- Часові межі для виконання процесів чіткіші й більш передбачувані, ніж у системах загального призначення:

  Завдання мають завершитися (розпочатися) до деякого дедлайну.

  (Або всі й обов'язково, або якомога більше)

- Суворіше використання пріоритетів та витіснення (щоб відповідати вимогам реального часу)
- **Не підходить:** FIFO
- **Не підходить:** пріоритетне планування у чистому вигляді
- **Підходить:** поєднання пріоритетів з перериваннями на основі таймера
- Може мати місце **негайне переривання** (*immediate preemtion*):
  ОС відповідає на переривання якомога швидше (майже негайно).
  Виняток - коли ОС виконує критично важливий код (має бути виконаний до кінця).

3. Приклади алгоритмів планування

Основні підходи до планування у системах реального часу

Статичний підхід на основі таблиці (static table-driven approach)
- Статичний аналіз підходящого процесу.
- Статично: коли процес має розпочатися.

Статичний підхід з пріоритетами і витісненням (static priority-driven preemptive)
- Статичний аналіз підходящого процесу.
- Статично: пріоритети процесів.

Статичний підхід - добрий для періодичних завдань.
Якщо зміни - доведеться перероблювати весь розклад.

3. Приклади алгоритмів планування

Основні підходи до планування у системах реального часу

Динамічний підхід на основі планування (dynamic planning-based approach)

- Вибір підходящого процесу - переважно у ході роботи ОС.
- Новоприбуле завдання приймається до виконання тільки коли відповідає часовим обмеженням (коли його можна поставити у розклад).

Динамічний підхід негарантованого виконання (dynamic best effort approach, динамічний підхід найкращих зусиль)

- Аналіз того, наскільки процес підходящий, не робиться.
- Головне - дотриматися дедлайнів.
- Якщо процес не встигає до дедлайну - його буде завершено.
- Процесу присвоюється пріоритет (за його характеристиками)

Для самостійного читання

1. **[*Silberschatz, Galvin, Gagne, 2018*]** Chapter 5.
2. **[*Stollings, 2017*]** Chapter 9-10.
3. **[*Tanenbaum, Bos, 2014*]** Chapter 2 (paragr. 2.4).
4. **[*Шеховцов, 2009*]** Розділ 4.