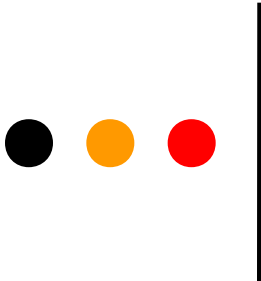




Операційні системи

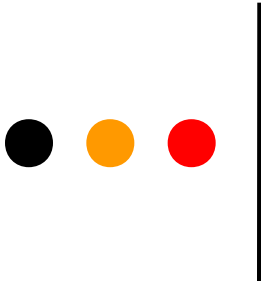
Лекція 5

Керування процесами і потоками
(продовження) – планування



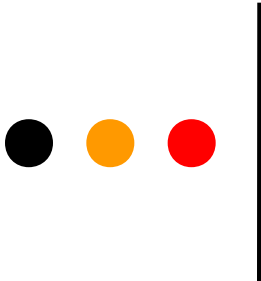
План лекції

- Завдання планування
- Витісняльна і невитісняльна багатозадачність
- Приоритетні і безпріоритетні дисципліни планування
- Квантування
- Алгоритми планування
- Керування процесами і потоками у сучасних ОС
 - UNIX
 - Linux
 - Solaris
 - Windows



Планування виконання процесів або потоків

- **Планування** (*scheduling*) – розподіл процесорного часу між процесами або потоками
- Завдання планування:
 - Визначення моменту часу для зміни потоку, що виконується
 - Вибір наступного потоку для виконання
 - Переключення контекстів
- Перші два завдання вирішуються здебільшого програмними засобами, третє – головним чином, апаратними засобами із застосуванням **механізму переривань**



Витісняльні і невитісняльні алгоритми планування

- **Витісняльні** (*preemptive*) – рішення про переключення з виконання одного потоку на виконання іншого (“витіснення” потоку з процесора) приймає операційна система
- **Невитісняльні** (*non-preemptive*) – активний потік виконується, поки він сам за власною ініціативою не віддасть керування операційній системі
- Не плутати з пріоритетними/безпріоритетними дисциплінами планування!



Квантування

- Зміна потоку відбувається, якщо:
 - Потік завершився
 - Виникла помилка (переривання)
 - Потік перейшов у стан очікування
 - *Вичерпано квант процесорного часу*
- Квантування – це один з підходів, що реалізує витісняльну багатозадачність
- Кванти можуть бути фіксованої величини, або змінюватись




Пріоритети

- Розрізняють **пріоритетні** і **безпріоритетні дисципліни планування**
- **Відносні пріоритети** впливають лише на вибір процесу з черги на виконання
 - Або завжди вибирають лише процес з найвищим пріоритетом
 - Або обслуговуються усі черги, але пропорційно пріоритетам (алгоритм **Weighed Round Robin, WRR**)
- **Абсолютні пріоритети** – це один з алгоритмів реалізації витісняльної багатозадачності. Зміна потоку:
 - Потік завершився
 - Виникла помилка (переривання)
 - Потік перейшов у стан очікування
 - У черзі з'явився потік з вищим пріоритетом



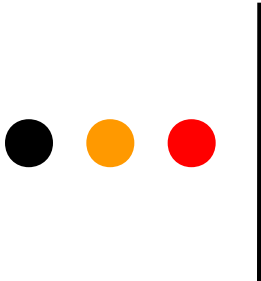
Алгоритми планування

- **Планування за принципом FIFO**
 - Одна черга потоків
 - Невитісняльне планування
 - Проблема – “**ефект конвою**”
- **Кругове планування (round-robin scheduling)**
 - Безпріоритетне планування (одна черга потоків)
 - Квантування у “чистому” вигляді
 - Рекомендована довжина кванта – 10-100 мс
- **Багаторівневі черги (multilevel queues)**
 - Кілька черг для груп потоків із різними пріоритетами
 - Якщо в черзі немає жодного потоку, переходять до черги з нижчим пріоритетом
 - У кожній черзі застосовують простий алгоритм (наприклад, кругове планування), не звертаючи уваги на потоки в інших чергах
 - Для різних черг можна застосовувати різні алгоритми
 - Проблема – “**голодування**” (**starvation**)



Планування на підставі характеристик подальшого виконання

- **STCF** (*Shortest Time to Completion First, перший – із найкоротшим часом виконання*)
 - Алгоритм є теоретично оптимальним за критерієм середнього часу відгуку
 - Недолік – не завжди можна передбачити час виконання (підходить для довготермінового планування, не підходить для короткотермінового)
 - Аналог з витісняльним плануванням (за принципом абсолютних пріоритетів) – **SRTCF** (*Shortest Remaining Time to Completion First, перший – із найкоротшим часом виконання, що залишився*)
- **Багаторівневі черги зі зворотним зв'язком** (*Multilevel Feedback Queues*)
 - Потокам дозволено переходити з черги в чергу
 - Потоки у черзі об'єднуються не за пріоритетом, а за довжиною інтервалу використання процесора
 - Коли потік не вичерпав квант – він становиться у кінець тієї ж черги, коли вичерпав – переводиться у нижчу чергу



Лотерейне планування (*lottery scheduling*)

- Принцип:
 - Потік отримує певну кількість “лотерейних квитків”, кожен з яких дає право користуватись процесором упродовж часу T
 - Планувальник через проміжок часу T проводить “розіграш”; потік, що “виграв”, дістає керування
- Шляхом розподілу і динамічного перерозподілу квитків можна:
 - Емулювати кругове планування
 - Емулювати планування з пріоритетами
 - Емулювати CRTCF
 - Забезпечити заданий розподіл процесорного часу між потоками
 - Динамічно змінювати пріоритети



Планування в UNIX SVR4

- Поняття “потік” відсутнє, планування здійснюється для процесів
- Реалізована витісняльна багатозадачність, що заснована на пріоритетах і квантуванні
- Визначені три пріоритетні класи (кожний процес належить до одного з класів):
 - **Реального часу**
 - Фіксовані пріоритети, але користувач може їх змінювати
 - Характеристики: рівень глобального пріоритету і квант часу
 - За наявності готових до виконання процесів реального часу інші процеси взагалі не розглядаються
 - **Системних процесів**
 - Зарезервований для ядра системи
 - Рівень пріоритету призначається ядром і ніколи не змінюється
 - **Розподілу часу**
 - Цей клас призначається новому процесу за умовчанням
 - Пріоритет обчислюється з двох складових: користувацької і системної
 - Користувацьку частину можуть змінювати адміністратор (в обидва боки) і власник процесу (лише у бік зниження пріоритету)
 - Системну частину змінює планувальник: знижує пріоритет процесам, що не уходять в стан очікування, підвищує пріоритет процесам, що часто уходять в стан очікування



Планування в Solaris

- Порівняно з SVR4 введені додаткові пріоритетні класи
 - **System (SYS)** – лише для потоків ядра
 - **Realtime (RT)** – клас з найвищим пріоритетом (вищим за **SYS**)
 - **Timeshare (TS)** – традиційний клас розподілу часу
 - **Interactive (IA)** – “інтерактивний” клас, призначений для потоків, що пов'язані з віконною системою
 - Передбачене підвищення пріоритету вікна, що знаходиться у фокусі
 - Той самий діапазон пріоритетів, і та ж таблиця диспетчеризації, що й для **TS**
 - **Fair Share Scheduler (FSS)** – клас, що контролюється не динамічним пріоритетом, а виділеними й наявними ресурсами ЦП
 - Ресурси ЦП поділяються на частки (**shares**), які виділяє адміністратор
 - Той же діапазон пріоритетів, що й для **TS/IA**
 - **Fixed Priority (FX)** – клас, для якого ядро не змінює пріоритет
- Ядро завантажує планувальники як модулі, паралельно можуть діяти різні планувальники



Планування в Solaris (2)

- Ядро підтримує окремі черги (фактично, черги черг) для кожного процесора
 - Черги впорядковуються за пріоритетами потоків
 - Зайнятість черг подається у вигляді бітової маски
- Для потоків реального часу підтримується єдина системна черга (не поділяється на процесори, але може розподілятися на групи процесорів (*processor sets*), якщо такі призначені)
- Кожен потік має 2 пріоритети: *глобальний*, що виводиться з його пріоритетного класу, і *успадкований* пріоритет
- Здебільшого, потоки виконуються як потоки класів **TS** або **IA**
 - Потоки **IA** створюються, коли потік асоціюється з віконною системою
 - Потоки **RT** створюються явно
 - Потоки **SYS** – це потоки ядра
- Переривання мають пріоритет ще вищий, ніж потоки **RT**



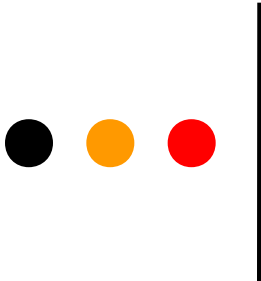
Планування в Linux

- Як і в UNIX, планування здійснюється **для процесів**
- Визначені три групи процесів у системі:
 - Реального часу із плануванням за принципом FIFO
 - Реального часу із круговим плануванням
 - Звичайні
- Особливості планування процесів реального часу
 - Вони завжди мають пріоритет перед звичайними процесами
 - Процеси із плануванням за принципом FIFO або самі віддають процесор, або їх витісняють процеси реального часу з більшим пріоритетом
 - Процеси із круговим плануванням додатково витісняють по завершенні кванту часу



Традиційний алгоритм планування в Linux (до 2.4)

- Процесорний час поділяється на **епохи**
- У кожній епосі процес має квант часу, який розраховують на початку епохи (**базовий квант**)
- Епоха закінчується, коли усі готові до виконання процеси вичерпали свої кванти
- Значення кванту може змінюватись системними викликами **nice()** і **setpriority()**
- Пріоритет буває **фіксований** (для процесів реального часу) і **динамічний** (для звичайних процесів)
- Динамічний пріоритет залежить від
 - **Базового пріоритету** (**nice** – задає величину, на якій ґрунтується базовий квант процесу)
 - **Часу, що залишився** до вичерпання кванту (**counter** – кількість переривань таймера, на початку епохи йому надають значення базового кванту і зменшують на одиницю в обробнику переривань таймера)



Планування в Linux з ядром 2.6 (3, 4)

- Недоліки традиційного алгоритму
 - Значний час на розрахунки на початку кожної епохи
 - Значний час на обрання процесу (потрібний розрахунок динамічного пріоритету)
 - Одна черга процесів – погано пристосовується для багатопроцесорних систем
- Новий підхід
 - На кожний процесор – своя черга
 - Кожна черга готових процесів – це масив черг готових процесів, де елементи упорядковані за динамічним пріоритетом
 - Процеси, що вичерпали квант, переносять в інший масив – масив черг процесів, що вичерпали квант
 - По завершенні епохи масиви міняють місцями



Планування у Windows

- Планування здійснюється виключно **для потоків** (ядро не розрізняє, яким процесам належать потоки)
- Під час планування ядро працює з блоками **KTHREAD**
- Пріоритети (від 1 до 31, динамічні – від 1 до 15):
 - **Real-time** ~ 24
 - **High** ~ 13
 - **Normal** ~ 8
 - **Idle** ~ 4
- Відносні пріоритети потоків від -2 до +2 від базового
- Кванти
 - **Короткі кванти змінної довжини** (10 або 30 мс) – перевага інтерактивних застосунків
 - **Довгі кванти фіксованої довжини** (120 мс) – перевага фонових процесів
- Список готових потоків складається з 31 елементу (відповідно до рівнів пріоритетів), з кожним з яких пов'язана черга
- Динамічна зміна пріоритету і кванту часу: **підтримка** (**boosting**) і **ослаблення** (**decay**), запобігання **голодуванню**



Операційні системи реального часу Real-time Operating Systems (RTOS)

- Стандарт POSIX 1003.1 дає визначення:
«Реальний час в операційних системах – це здатність операційної системи забезпечити потрібний рівень сервісу у визначений проміжок часу»
- В якості головної вимоги до **RTOS** висувають вимогу забезпечення передбачуваності або детермінованості поведінки системи у найгірших зовнішніх умовах
 - Це суттєво відрізняється від вимог до продуктивності та швидкодії універсальних ОС
- В системах реального часу необхідне введення деякого директивного терміну (**deadline**), до завершення якого задача повинна бути виконана обов'язково
 - Цей директивний термін застосовується планувальником завдань як для призначення пріоритету завдання під час його запуску, так і під час вибору завдання на виконання



Системи жорсткого реального часу і м'якого реального часу

- Розрізняють системи **жорсткого** (*hard*) реального часу і **м'якого** (*soft*) реального часу
- В системах жорсткого реального часу нездатність забезпечити реакцію на деякі події протягом заданого часу призводить до відмов і неможливості виконання поставленого завдання
 - Такі системи називають системами **з детермінованим часом**
- Системами м'якого реального часу називають системи, що не підпадають під визначення "жорсткі"
 - В літературі чіткого визначення для них поки немає
 - Системи м'якого реального часу намагаються забезпечити реакцію на зовнішні події протягом заданого часу, але вони можуть не встигати зробити це, і це не призводить до відмови системи у цілому



Планування в RTOS

- Головною проблемою в RTOS є планування (**scheduling**), яке має забезпечити передбачувану поведінку системи за усіх обставин
 - Процес з дедлайнами повинен стартувати й виконуватись так, щоби він не пропустив жодного свого дедлайну
 - Якщо це неможливо, процесу має бути відмовлено у запуску
- Вивчають і розвивають два підходи:
 - **статичні** алгоритми планування (**RMS – Rate Monotonic Scheduling**)
 - **динамічні** алгоритми планування (**EDF – Earliest Deadline First**)



RMS vs EDF

- RMS застосовують для формального доведення умов передбачуваності системи
 - Для реалізації цього необхідно **планування на основі абсолютних пріоритетів**, тобто пріоритетів, що переривають обслуговування (**preemptive priority scheduling**)
 - В теорії RMS пріоритет заздалегідь призначають кожному процесу
 - Процеси повинні задовольняти таким умовам:
 - процес має бути завершеним за час його періоду,
 - процеси не залежать один від одного,
 - кожному процесу необхідний однаковий процесорний час на кожному інтервалі,
 - у неперіодичних процесів немає жорстких термінів,
 - переривання процесу відбувається за обмежений час.
 - Процеси виконують відповідно до пріоритетів
 - Перевага надається задачам з **найкоротшими періодами виконання**
- В EDF пріоритет призначають динамічно, і найбільший пріоритет надають процесу, в якого залишився **найменший час виконання**
 - За великого завантаження системи EDF має деякі переваги перед RMS