12.5 АЛЬТЕРНАТИВНІ СТРАТЕГІЇ ПЛАНУВАННЯ

У табл. 12.1 представлена інформація про різні стратегії планування, які будуть розглянуті нижче.

Функція вибору процесу визначає, який з готових до виконання процесів буде обраний наступним для виконання. Функція може бути заснована на пріоритеті, вимогах до ресурсів або характеристиках виконання процесів. При цьому мають значення такі параметри:

– w – час, витрачений на цей момент системою (очікування і виконання);

– e – час, витрачений на цей момент на виконання;

– s – загальний час обслуговування, що потрібний процесу, включаючи e (зазвичай ця величина оцінюється або задається користувачем).

Наприклад, вибір функцій max[w] визначає стратегію «першим прийшов, першим обслужений» (first come first served – FCFS).

Таблиця 12.1. Характеристики різних стратегій планування

Стратегія планування	**Функція вибору**	**Режими виконання**	**Пропускна здатність**	**Голодування**
FCFS	max[w]	Невитісняючі	Не важливо	Немає
Кругова (RR)	const	Витісняючі (за часом)	Низька при малому кванті часу	Немає
SPN	min[s]	Невитісняючі	Висока	Можливо
SRT	min[s-e]	Витісняючі (за рішенням)	Висока	Можливо
HRRN	max[(w+s)/s]	Невитісняючі	Не важливо	Немає

Режим рішення визначає, в які моменти часу виконується функція вибору. Режими рішення діляться на дві категорії.

1. **Невитісняючі**. У цьому випадку процес, який знаходиться в стані виконання, продовжує виконання до тих пір, поки він не завершиться або не опиниться в заблокованому стані.

2. **Витісняючі**. Процес, що виконується в даний момент, може бути перерваний і переведений ОС в стан готовності до виконання. Рішення про витіснення може прийматися при запуску нового процесу по перериванню, яке

призводить заблокований процес в стан готовності, або періодично – на основі переривань таймера.

Витісняючі стратегії призводять до підвищених накладних витрат у порівнянні з невитісняючими, але при цьому забезпечують кращий рівень обслуговування всієї множини процесів, оскільки запобігають монопольному використанню процесора протягом тривалого часу одним з процесів. Крім того, використання ефективних механізмів перемикання процесів і великий обсяг основної пам'яті дозволяє підтримувати відносно невелику вартість витіснення.

Розглянемо ці алгоритми планування процесів.

Першим поступив – першим обслужений (FCFS). Найпростіша стратегія планування «першим прийшов, першим обслужений» (first come first served – FCFS) відома також як схема строгої черговості. Як тільки процес стає готовим до виконання, він приєднується до черги готових процесів. При припиненні виконання поточного процесу для виконання вибирається процес, що знаходиться в черзі довше інших. Черга подібного типу має в програмуванні спеціальне найменування – FIFO (скорочення від First In, First Out – першим увійшов, першим вийшов). Такий алгоритм вибору процесу здійснює невитісняюче планування (рис. 12.10).

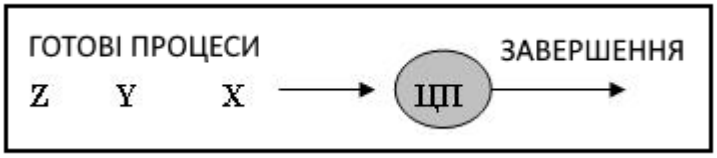
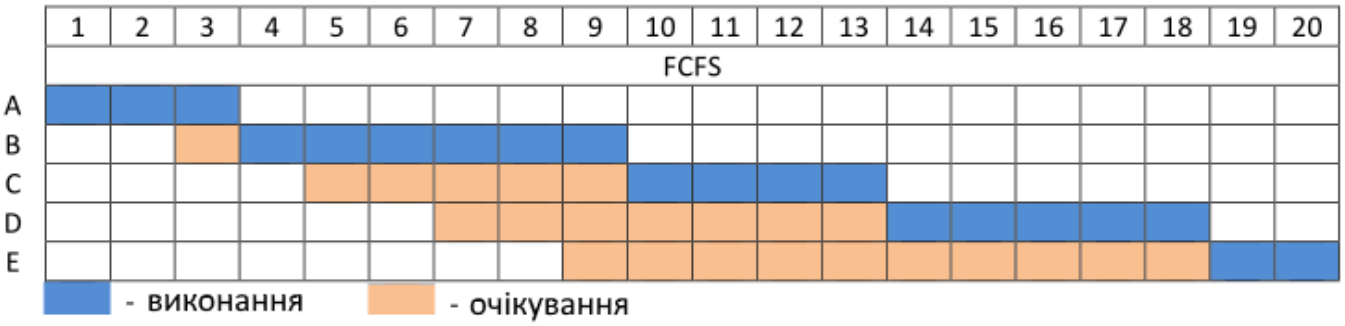


Рисунок 12.10 – Планування за принципом FIFO

Перевагою алгоритму FCFS є легкість його реалізації, але в той же час він має багато недоліків. Стратегія FCFS набагато краще працює для довгих процесів, ніж для коротких. При роботі з процесами, орієнтованими на роботу з процесором, такі процеси отримують перевагу над процесами, орієнтованими на введення-виведення. Для однопроцесорних систем FCFS – це не найкраща стратегія, але вона часто комбінується з використанням пріоритетів. В цьому випадку планувальник підтримує ряд черг, по одній для кожного рівня пріоритету, і працює з процесами в кожній черзі за стратегією FCFS.

Розглянемо наступний приклад. Нехай в стані готовність знаходяться п'ять процесів (A, B, C, D, E), для яких відомий час появи в черзі готових процесів (0, 2, 4, 6, 8) і час їх виконання (обслуговування, CPU – 3, 6, 4, 5, 2). Ці часи наведені в деяких умовних одиницях.

Будемо вважати, що вся діяльність процесів обмежується використанням лише одного проміжку CPU, що процеси не роблять операцій введення- виведення, не будемо брати також до уваги і час перемикання контексту. Якщо процеси розташовані в черзі процесів, готових до виконання, в порядку A, B, C, D, E, то картина їх виконання виглядає таким чином (рис. 12.11).



Процес	A	B	C	D	E	Середній час
Час появи в черзі	0	2	4	6	8	
Час виконання	3	6	4	5	2	
Час завершення	3	9	13	18	20	
Час обороту	3	7	9	12	12	8.60
Час очікування	0	1	5	7	10	4.60
Нормалізований повний час	1.00	1.17	2.25	2.40	6.00	2.56

Рисунок 12.11 – Виконання процесів алгоритмом FCFS

Час очікування для процесу A становить 0 одиниць часу, для процесу B – 1, для процесу C – 5, для процесу D – 7, для процесу E – 10. Таким чином, середній час очікування в цьому випадку – (0 + 1 + 5 + 7 + 10)/5 = 4.60 одиниць часу.

Повний час виконання (час обороту), витрачений процесом в системі (час очікування + час обслуговування), для процесу A становить 3 одиниці часу, для процесу B – 1 + 6 = 7 одиниць, для процесу C – 5 + 4 = 9 одиниць, для процесу D – 7

+ 5 = 12 одиниць, для процесу E – 10 + 2 = 12 одиниць. Середній повний час виконання виявляється рівним (3 + 7 + 9 + 12 + 12)/5 = 8.60 одиницям часу.

Кориснішою величиною є **нормалізований повний час**, який визначається як відношення повного часу до часу виконання і вказує відносну затримку, що випробовується процесом. Мінімальне значення цього відношення – 1. Зростання значення відповідає зниженню рівня обслуговування.

На рис. 12.11 видно, що алгоритм FCFS набагато краще працює для довгих процесів, чим для коротких. Адже для коротких процесів час очікування може істотно перевищувати час обслуговування.

Кругове планування. Очевидний шлях підвищення ефективності роботи з короткими процесами в схемі FCFS – використання витіснення на основі таймера. Найпростіша стратегія, заснована на цій ідеї, – стратегія кругового (карусельного) планування (round robin – RR). Таймер генерує переривання через певні інтервали часу. При кожному перериванні процес, що виконується в даний момент, поміщається в чергу готових до виконання, і починає виконуватися черговий процес, який обирається відповідно до стратегії FCFS. Ця методика відома також як квантування часу, оскільки перед тим як опинитися витісненим, кожен процес отримує квант часу для виконання (рис. 12.12).

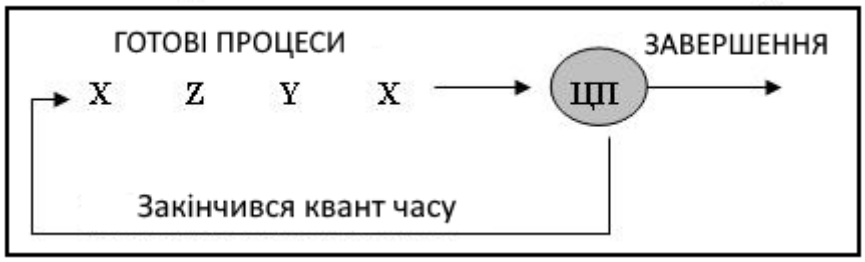


Рисунок 12.12 – Планування за принципом RR

При круговому плануванні важливим є питання про тривалість кванта часу. Так короткі процеси будуть відносно швидко проходити через систему, але з істотними накладними витратами, пов'язаними з обробкою переривань і виконанням функцій планувальника. Кругова стратегія ефективна в системах загального призначення з розподілом часу.

Таймер генерує переривання через певні інтервали часу. При кожному перериванні процес, що виконується, поміщається в чергу готових до виконання

процесів, і починає виконуватися черговий процес, вибираний відповідно до стратегії FCFS. Ця методика відома також як «квантування часу» (time slicing), оскільки кожен процес отримує кванту часу для виконання (рис. 12.13) [22].

При круговому плануванні принциповим стає питання про тривалість кванта часу. При дуже великих величинах кванта часу, коли кожен процес встигає завершити свою роботу до виникнення переривання за часом, алгоритм RR вироджується в алгоритм FCFS. При дуже малих величинах створюється ілюзія того, що кожен з n процесів працює на власному віртуальному процесорі з продуктивністю ~ 1/n від продуктивності реального процесора. Правда, це справедливо лише при теоретичному аналізі за умови нехтування часом перемикання контексту процесів. У реальних умовах при занадто малій величині кванта часу і, відповідно, занадто частому перемиканні контексту накладні витрати на перемикання різко знижують продуктивність системи.

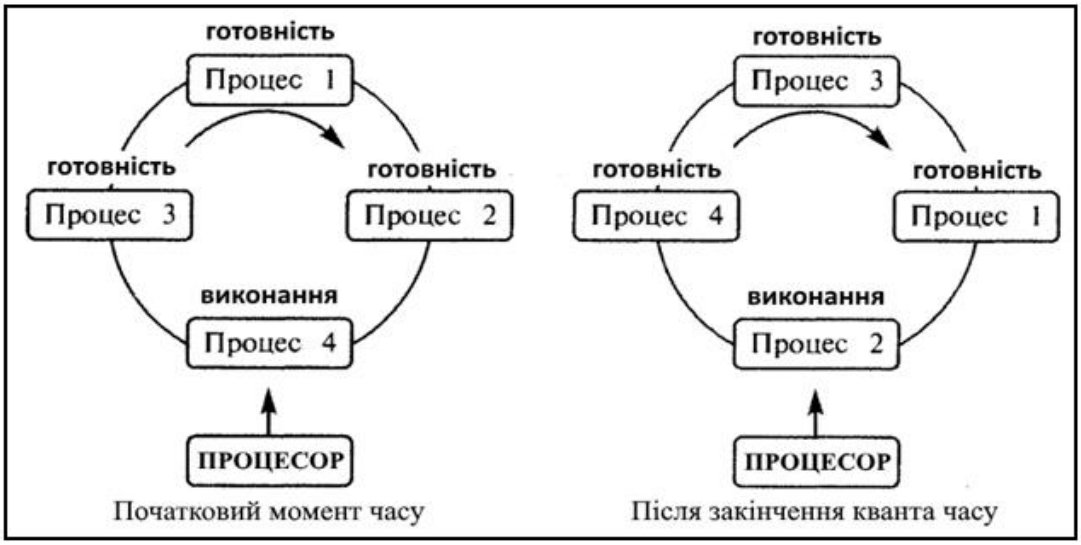


Рисунок 12.13 – Кругове планування

Розглянемо попередній приклад з тим же порядком процесів, часом появи в черзі готових процесів, часом їх виконання і величиною кванта часу q=1. Виконання цих процесів ілюструється рис. 12.14.

На рис. 12.15 показані результати роботи RR алгоритму при використанні кванта часу q з тривалістю 4 одиниці часу. Зверніть увагу, що найбільш короткий процес E значно швидше проходить через систему при малому кванті часу.

Процес	A	B	C	D	E	Середній час
Час появи в черзі	0	2	4	6	8	
Час виконання	3	6	4	5	2	
Час завершення	4	18	17	20	15	
Час обороту	4	16	13	14	7	10.80
Час очікування	1	10	9	9	5	6.80
Нормалізований повний час	1.33	2.67	3.25	2.80	3.50	2.71

Рисунок 12.14 – Виконання процесів алгоритмом RR з квантом часу q=1

Процес	A	B	C	D	E	Середній час
Час появи в черзі	0	2	4	6	8	
Час виконання	3	6	4	5	2	
Час завершення	3	17	11	20	19	
Час обороту	3	15	7	14	11	10.00
Час очікування	0	9	3	9	9	6.00
Нормалізований повний час	1.00	2.50	1.75	2.80	5.50	2.71

Рисунок 12.15 – Виконання процесів алгоритмом RR з квантом часу q=4

Вибір найкоротшого процесу. Ще один шлях до зниження перекосу на користь довгих процесів – використання стратегії вибору найкоротшого процесу (shortest process next - SPN). У літературі зустрічається і інша назва: «найкоротша робота першою» або Shortest Job First (SJF). Це не витісняюча стратегія, при якій для виконання вибирається процес з найменшим очікуваним часом виконання.

Основні труднощі в застосуванні стратегії SPN полягають у тому, як оцінити час виконання кожному процесу. Для пакетних завдань це може зробити програміст, а для виконання інтерактивних процесів операційна система підраховує час виконання за спеціальними формулами.

Основний ризик при використанні стратегії SPN полягає в можливому голодуванні довгих процесів при стабільній роботі коротких процесів. Його застосування небажане в системах з розподілом часу або системах обробки транзакцій через відсутність витіснення.

На рис. 12.16 приведені результати застосування цього алгоритму до нашого прикладу. Зверніть увагу, що процес E обслуговується набагато раніше, ніж у разі застосування алгоритму FCFS. Відносно часу відгуку загальна продуктивність системи також зростає, але при цьому збільшується розкид його величини, особливо для довгих процесів, і, відповідно, знижується передбачуваність.

Процес	A	B	C	D	E	Середній час
Час появи в черзі	0	2	4	6	8	
Час виконання	3	6	4	5	2	
Час завершення	3	9	15	20	11	
Час обороту	3	7	11	14	3	7.60
Час очікування	0	1	7	9	1	3.60
Нормалізований повний час	1.00	1.17	2.75	2.80	1.50	1.84

Рисунок 12.16 – Виконання процесів алгоритмом SPN

Основний ризик при використанні алгоритму SPN полягає в можливому голодуванні довгих процесів при стабільній роботі коротких процесів.

Найменший час, що залишається. Стратегія найменшого часу, що залишається (shortest remaining time – SRT) являє собою витісняючу версію стратегії SPN. У цьому випадку планувальник вибирає процес з найменшим очікуваним часом до закінчення процесу. При приєднанні нового процесу до черги готових до виконання процесів може виявитися, що його час, що залишився, менше, ніж час, що залишився у виконуваного в даний момент процесу. Планувальник, відповідно, може застосувати витіснення при готовності нового процесу. Як і при використанні стратегії SPN, планувальник для коректної роботи функції вибору повинен оцінити час виконання процесу. У цьому випадку також є ризик голодування довгих процесів.

У разі використання алгоритму SRT немає таких великих перекосів на користь довгих процесів, як при використанні алгоритму FCFS. У відмінності від алгоритму RR, тут не генеруються додаткові переривання, що знижує накладні витрати. Проте, в цьому випадку відбувається збільшення накладних витрат із- за необхідності фіксувати і записувати час виконання процесів. У зв'язку з тим що короткі завдання негайно отримують перевагу перед довгими завданнями, що виконуються, алгоритм SRT істотно виграє в алгоритму SPN в часі обороту. На рис. 12.17 приведені результати застосування цього алгоритму до нашого прикладу.

Процес	A	B	C	D	E	Середній час
Час появи в черзі	0	2	4	6	8	
Час виконання	3	6	4	5	2	
Час завершення	3	15	8	20	10	
Час обороту	3	13	4	14	2	7.20
Час очікування	0	7	0	9	0	3.20
Нормалізованний повний час	1.00	2.17	1.00	2.80	1.00	1.59

Рисунок 12.17 – Виконання процесів алгоритмом SRT

В даному прикладі три найбільш коротких процеси обслуговуються негайно, що призводить до нормалізованого часу обороту для кожного з них, рівному 1.00.

Планування за найбільшим відносним часом реакції. Брінч Хансен розробив стратегію планування за найбільшим відносним часом реакції (highest response ratio next – **HRRN**), який компенсує деякі недоліки алгоритму SPF (найкоротший процес першим): надмірне упередження проти довгих процесів і надмірну прихильність до коротких нових процесів. HRRN – це дисципліна планування без пріоритетного витіснення, згідно з якою пріоритет кожного процесу є не тільки функцією часу обслуговування цього процесу, але також часу, витраченого процесом на очікування обслуговування. Після того як процес отримає у своє розпорядження процесор, він виконується до завершення. Динамічні пріоритети при дисципліні обслуговування HRRN обчислюються за формулою:

$$\mathbf{P = (W + S)/S,}$$

де **P** – пріоритет;

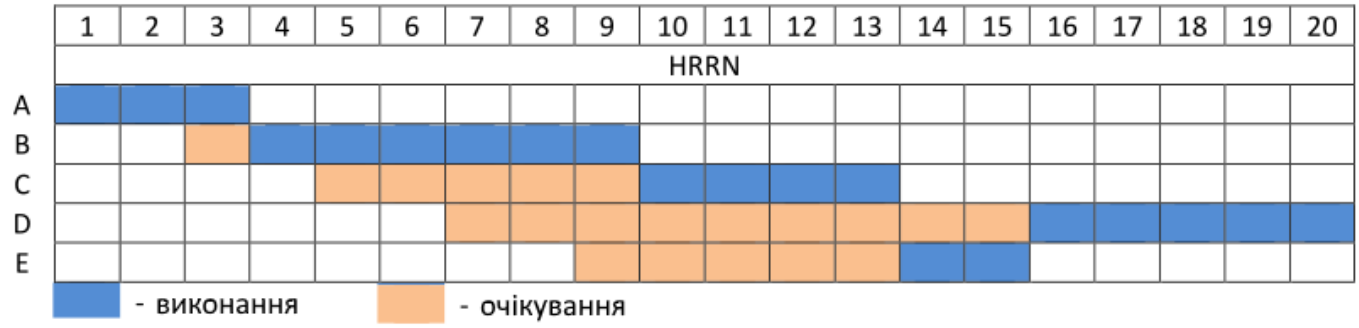
W – час, витрачений процесом на очікування;

S – очікуваний час обслуговування.

Зауважимо, що процес при вході в систему отримує мінімальне значення **P** (рівне 1.0). Таким чином, правило стратегії планування найвищого відносного часу реакції (відгуку) можна сформулювати так: при завершенні або блокуванні поточного процесу для виконання з черги готових процесів вибирається той, який має найбільше значення P.

Приклад роботи алгоритму HRRN представлений на рис. 12.18. Як і у разі алгоритмів SRT і SPN, в описаному алгоритмі потрібна оцінка часу обслуговування для визначення максимального значення P.

Оскільки час виконання S знаходиться в знаменнику, та перевага виявлятиметься коротшим процесам. Проте, оскільки в чисельнику є час очікування W, довші процеси, які вже досить довго чекають, також отримуватимуть певну перевагу. На практиці пріоритети перераховуються через певні проміжки часу, і відповідно до змін черга переупорядковується.



Процес	A	B	C	D	E	Середній час
Час появи в черзі	0	2	4	6	8	
Час виконання	3	6	4	5	2	
Час завершення	3	9	13	20	15	
Час обороту	3	7	9	14	7	8.00
Час очікування	0	1	5	9	5	4.00
Нормалізованний повний час	1.00	1.17	2.25	2.80	3.50	2.14

Рисунок 12.18 – Виконання процесів алгоритмом HRRN