

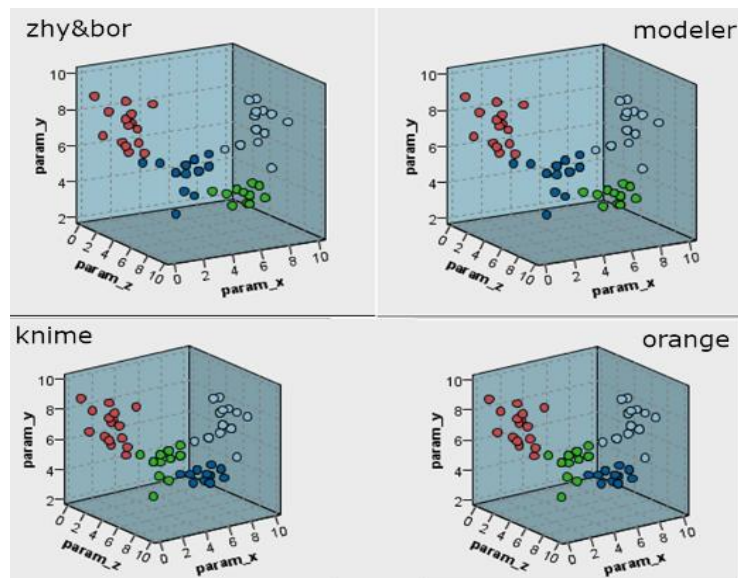


Рис.6 – Візуалізація результатів кластеризації (4 кластери)

Висновки

1. Найкращі результати оцінки якості кластеризації по більшості індексів отримані за допомогою Zhy&Bor та Modeler. Це дає підстави стверджувати, що індекси відпрацьовують коректно і їх можна використовувати для проведення подальших досліджень.
2. Zhy&Bor має перевагу перед іншими програмними продуктами, тому що він є окремим модулем, який можна інтегрувати з іншими системами, також в ньому реалізований автоматичний розрахунок кількості кластерів.

СПИСОК ЛІТЕРАТУРИ

- [1] M.Halkidi, Y. Batistakis, M. Vazirgiannis, “On clustering validation techniques”, *Journal of intelligent information systems*, vol. 17, no.2-3, pp.107-145, 2001.
- [2] Е.В. Сивоголовко, “Методы оценки качества чёткой кластеризации”, *Компьютерные инструменты в образовании*, №4, с.14-31, 2011.

ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ В ОПЕРАЦІЙНИХ СИСТЕМАХ ШЛЯХОМ ВИРІШЕННЯ КОНФЛІКТНИХ СИТУАЦІЙ МІЖ процесами

Завертайло К.С., Хошаба О.М.

Однією із основних проблем в операційних системах є виникнення взаємних блокувань між процесами. Дана проблема значно знижує продуктивність роботи операційних систем. Для вирішення даної проблеми було запропоновано багато методів і розроблено ряд алгоритмів, але конкуренція процесів всеодно породжує взаємні блокування.

В даній роботі розглянуто основні методи запобігання взаємним блокуванням їх переваги, недоліки і застосування на практиці. Також розглянуто методи виявлення та усунення взаємних блокувань в операційних системах і алгоритми, що розроблені на базі вказаних методів. Піднято питання швидкості виявлення взаємних блокувань в операційних системах. Розглянуто існуючі методи виявлення взаємних блокувань. Запропоновано новий спосіб підвищення швидкості виявлення взаємних блокувань.

One of the main problems in operating systems is the occurrence of deadlocks between processes. This problem significantly reduces the performance of operating systems. To solve this problem, many methods have been proposed and a number of algorithms have been developed, but the competition of processes equally generates deadlocks.

This paper discusses the main methods of preventing deadlocking, their advantages, disadvantages and practical application. Methods for detecting and eliminating deadlocks in operating systems and algorithms for crushing based on these methods are also considered. The question of the speed of detecting deadlocks in operating systems was raised. Existing methods for detecting deadlocks are considered. A new way to improve the rate of deadlock detection is proposed.

На сьогоднішній день одне з найважливіших питань в інформаційних технологіях – це питання продуктивності роботи операційних систем. Однією з проблем, яка негативно впливає на продуктивність операційних систем є проблемою виникнення взаємних блокувань. Взаємне блокування це ситуація коли два і більше процесу очікують ресурси зайнятими один одним [1]. Процесом в операційній системі є програма, що виконується в даний час, а ресурсом називається об'єкт до якого надається доступ.

Вище описана ситуація схематично зображена на рисунку 1 у вигляді трьох процесів та двох ресурсів, де процеси пронумеровані під цифрами 1, 2 і 3, а ресурси позначені літерами А і В. Ребро графа, яке вказує від ресурсу до процесу означає, що цей ресурс зайнятий вказаним процесом, а стрілка від процесу до ресурсу вказує на те, що даний процес запитує на використання даний ресурс і переходить у стан очікування до тих пір, поки процес, що утримує даний ресурс не звільнить його. Рисунок наочно демонструє те, що перший і другий процеси зайняли ресурси А і В відповідно, але не збираються їх звільнити. Процес 1 подає запит на використання ресурсу В, при цьому утримуючи ресурс А, а процес 2, навпаки, подає запит на використання ресурсу А, утримуючи ресурс В. З цього зрозуміло, що процеси не звільнять займані ними ресурси і будуть очікувати запит на використання ресурсів значний час, допоки не наступить певна зовнішня подія, яка цілком може бути завершенням роботи системи.

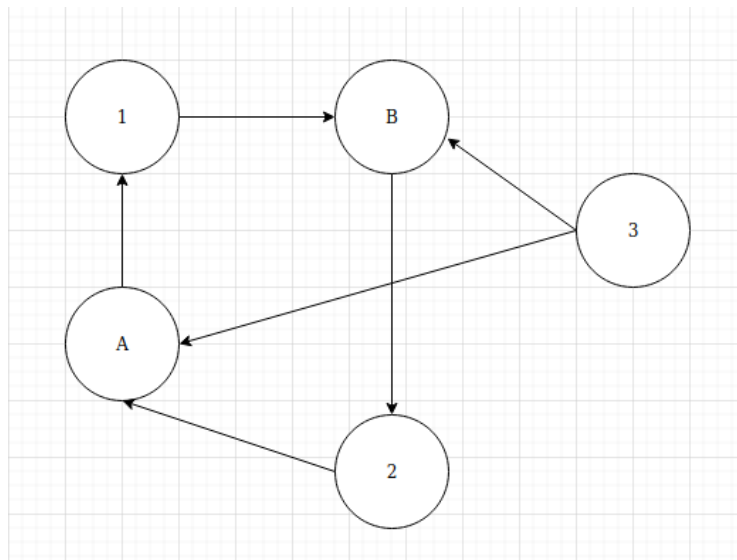


Рис. 1. Взаємне блокування

До описаної вище схеми спеціально додано ще один процес, який демонструє, що в такій ситуації можуть опинитись не тільки ті процеси, що утримують ресурси, а і ті, що взагалі не мають у своєму розпорядженні ніяких ресурсів [2]. Виникнення даної проблеми

викликає значне зниження продуктивності роботи операційної системи, оскільки запущені програми просто простоюють у стані очікування наданий планувальником час для них. Основними причинами виникнення описаної проблеми є конкуренція між процесами за ресурси у своїх цілях, зрозуміло, що кожен процес утримує ресурс і не буде його звільняти для іншого процесу; також виникненню цієї проблеми сприяє паралельне і квазі паралельне виконання запущених процесів. В даній роботі буде розглянуто існуючі методи боротьби з проблемою взаємних блокувань, методи упередження виникнення ситуації взаємних блокувань і швидкість виявлення тупикової ситуації (взаємного блокування) від моменту її виникнення.

В даній роботі розглядаються методи вирішення проблеми взаємних блокувань. Одним з таких методів є алгоритм страуса, тобто ігнорування проблеми. Звісно такий метод застосовується лише в тих операційних системах, де собі можна це дозволити, наприклад у системах де взаємні блокування виникають досить рідко. Це може бути система для якої час виконання поставлених завдань не є критичним, де процеси не часто займають ресурси у своїх цілях, або значна частина ресурсів може бути відібрана у певного процесу без певних збоїв у системі. Звичайно існує не мала кількість операційних систем для яких це не є вирішення проблеми, тому ігнорування даної проблеми не є її вирішенням.

Одними з методів вирішення проблеми взаємних блокувань є методи запобігання взаємним блокуванням за допомогою певного способу розподілу ресурсів між процесами в операційних системах. Одним з найпоширеніших і найдавніших є алгоритм банкіра, що був розроблений Едсгером Дейкстрою. Даний алгоритм розраховує відносно запитів чи є стан системи безпечним. Для реалізації цього алгоритму потрібно заздалегідь знати кількість запитів від кожного процесу в системі і кількість процесів в системі, що робить на практиці цей алгоритм непотрібним.

Не існує на сьогоднішній день доволі ефективних методів для запобігання взаємним блокуванням, оскільки майже всі методи. Як і алгоритм Дейкстри, потребують заздалегідь інформацію про кількість часу виділеного певному процесу, кількості процесів в операційній системі та кількості запитів від процесів у системі. Тому на практиці використовують методи, коли ситуація взаємного блокування вже виникла, саме через це в даній роботі і буде розглянуто основні методи усунення взаємних блокувань [5].

Один із методів виявлення взаємного блокування є метод виявлення взаємного блокування при використанні одного ресурсу кожного типу. Суттю метода є те, що будується граф у якому, зображені ресурси та процеси на вузлах. Якщо вузол у вигляді квадрату то це ресурс, а якщо у вигляді круга, то це процес. Коли ребро направлено від процесу до ресурсу, то це означає запит даного процесу на вказаний ним ресурс, а якщо ребро направлено від ресурсу до процесу, то це вказівка на те, який процес утримує даний ресурс. На рисунку 2 зображено граф, який схематично зображує роботу процесів в системі [4].

Послідовність роботи алгоритму дуже проста, береться певний процес за початкову точку і від нього по ребрам слідується до наступного вузла. Коли алгоритм проходиться весь до кінця і ніякі вузли не повторюються, то це означає, що взаємне блокування не виникло, тоді береться наступний вузол і так само проходиться весь граф. Якщо ж все ж таки попадаються повторення, то це означає, що було знайдено ситуацію взаємного блокування.

Як зрозуміло із назви метода він підходить для роботи такої системи, коли в неї є для кожного типу ресурсу лише один ресурс. Тому що повторитись можуть два різних ресурсу одного і того ж типу, а алгоритм зробить висновок про знайдене взаємне блокування.

Оскільки в більшості системах на сьогоднішній день є багато ресурсів одного і того ж типу, то необхідно розглянути метод виявлення взаємного блокування при використанні декількох ресурсів кожного типу. Даний метод виконується за допомогою двох матриць. На рисунку 3 зображено матрицю поточного розподілу[6]. У рядках матриці пронумеровані процеси, існуючи на даний момент в системі, а в стовпчиках – ресурси та кількість ресурсів

кожного типу, що є в системі. В самих ячейках зображено яку кількість ресурсів певного типу утримує в своєму розпорядженні кожен процес.

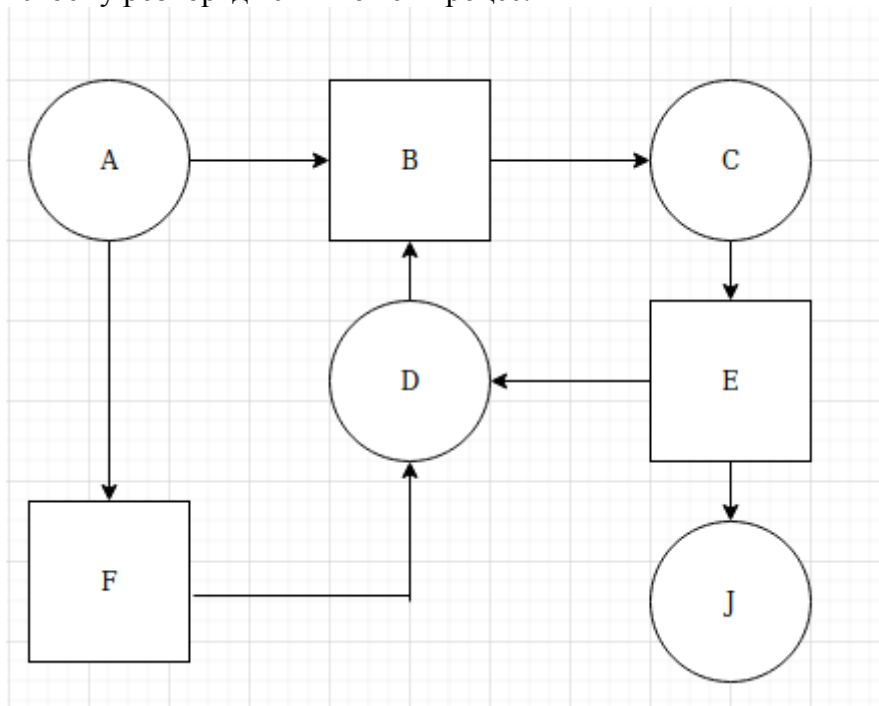


Рис. 2. Граф роботи процесів

Також для цього алгоритму вводиться ще одна матриця, яка відповідає за кількість запитів від кожного процесу, ця матриця так і називається матрицею запитів. Дана матриця зображена на рисунку 4. В ячейках цієї матриці зображено кількість запитів від кожного процесу на кількість ресурсів певного типу.

Робота самого алгоритму полягає в розгляді матриці запитів і виявлення того процесу, чий запити можуть бути повністю задоволені. Для того, щоб виявити кількість вільних для запиту ресурсів кожного типу, потрібно для цього порівнювати матрицю запитів і матрицю поточного розподілу, і виявляти процеси, запити яких можуть бути повністю задоволені. Таким чином перевіряються кожен процес в системі та знаходяться варіанти виходу із ситуацію [3]. Коли ж такі варіанти знайти неможливо, то така ситуація і є взаємним блокуванням.

Також дуже важливою є тема виходу із взаємного блокування, після того, як така ситуація була виявлена. Одним із найпоширеніших, але і найгрубіших, є простий спосіб знищення якогось процесу. Для цього просто проводиться огляд всіх процесів, що опинилися в заблокованому стані, і вибирається найменш значущий процес з найменшим пріоритетом і знищується. Звісно, такий метод є не зовсім сприйнятливим. Оскільки роботу певної задачі потрібно починати з самого початку. Також для виконання такої дії з конкретним процесом існує одна умова, потрібно оцінити наскільки погано це вплине на роботу всієї операційної системи, це нерідко залежить від самого ресурсу, який утримується процесом.

	2 рс	2 рс	5 рс	1 рс
1-й пр	0	0	2	1
2-й пр	0	2	0	0
3-й пр	1	0	2	1
4-й пр	0	0	3	0

Рис. 3. Матриця поточного розподілу

	1 рс	0 рс	2 рс	1 рс
1-й пр	0	1	0	0
2-й пр	1	0	2	0
3-й пр	0	1	0	1
4-й пр	0	0	1	0

Рис. 4. Матриця запитів

Також поширеним є спосіб пріоритетного володіння ресурсом. Суть способу полягає в тому, що береться конкретний процес, який утримує ресурс і цей ресурс в нього відбирається, всі дані процесу зберігаються поки йому не буде повернено даний ресурс. Але дане рішення також є не завжди прийнятливими, оскільки існують ресурси, які просто не можна примусово відібрати у процесу на порушивши роботу в цілому системи.

Вище було розглянуто найбільш поширені алгоритми та методи вирішення проблеми взаємного блокування, але дуже мало приділено уваги швидкості пошуку такої ситуації. Зазвичай в сучасних системах застосовується простий циклічний спосіб перевірки на наявність взаємного блокування через певний проміжок часу щоразу. В нього є певний недолік, оскільки найбільш часто така перевірка запускається марно, а зайві обчислення ніякій системі непотрібні. Тому в даній роботі буде запропоновано більш точний спосіб виявлення взаємного блокування в операційних системах.

Для того щоб більш точно обраховувати вірогідність виникнення взаємного блокування потрібно задати собі питання — які ознаки взаємного блокування проявляють процеси в операційній системі та задіяні поставленими завданнями перед процесами ресурси. Зрозуміло, що перша ознака цього це довгий простій певних процесів та довге утримання у своєму розпорядженні ресурсів цими ж процесами [8].

Якщо опиратися на вказану вище ознаку, то виявляти взаємні блокування система буде занадто пізно. Тобто тоді коли процеси та ресурси вже значний час знаходилися у стані простоювання і така ситуація вже встигла негативно вплинути на продуктивність

операційної системи. Тому для того щоб мати можливість якомога раніше виявляти взаємні блокування і при цьому не запускати цикл перевірки в пусту, потрібно оцінювати роботу процесів, їх взаємодію, та використання процесами ресурсів, у нормальному стані, коли ще конфліктів між процесами не виникло.

Перш за все потрібно спиратися на час роботи кожного процесу в операційній системі і час скільки той чи інший ресурс використовується процесами в системі. Для початку потрібно розглянути роботу ресурсів в операційній системі, а саме те, як використовують в своїх цілях їх процеси. Першим кроком з боку процесу по відношенню до ресурсу є запит на використання вказаного ресурсу. Коли ж ресурс зайнятий, то запит відхиляється і процес переходить у стан очікування того, коли даний ресурс звільниться. Якщо ресурс вільний, то процес отримує запитуваний ним ресурс в своє розпорядження.

Для більш точної оцінки того, коли потрібно запускати механізм перевірки на наявність взаємного блокування, потрібно створити окремий алгоритм, який буде працювати незалежно від роботи операційної системи і не буде ніяк на неї впливати. Звісно, що програмна реалізація такого алгоритму буде індивідуальна для кожної операційної системи, в залежності від роботи ядра, алгоритму планування та інших факторів. Для такої програми потрібно окремо зберігати дані, які будуть враховувати запити на використання ресурсів процесами, запити, які задовольняють вимоги процесів щодо ресурсів і звільнення ресурсів процесами. Дуже важливо фіксувати час того, коли кожен задіяний в системі ресурс був переданий в розпорядження конкретного процесу, та той час коли кожен займаний ресурс було звільнено процесом [9].

Сама суть запропонованого метода дуже проста, вона полягає в формулі 1, що зображена нижче.

$$t_r = t_2 - t_1; t_r \geq t_{r-n} * k \quad (1)$$

У вказаній формулі 1 t_r — це середнє арифметичне використання даного ресурсу всіма процесами, які його використовували від початку роботи операційної системи. t_1 — це момент часу, коли запит на використання конкретного ресурсу було задоволено тим процесом, що подавав запит. t_{12} — це момент часу, коли процес, що утримував вказаний ресурс, звільнив його і даний ресурс може використовуватися іншими процесами, що знаходяться в черзі. Також в даній формулі описано порівняння з попереднім значенням, що множиться на певний коефіцієнт в залежності від параметрів операційної системи та її стану, в якому вона знаходиться в даний час.

Суть формули дуже проста, вона дієва лише по відношенню до тих ресурсів, які часто використовуються в операційній системі запущеними процесами, але це є і перевага описаної вище формули, оскільки ресурси, які частіше використовуються процесами, мають більшу вірогідність потрапити у стан взаємного блокування. Таким чином після багатьох запитів буде виведене число, яке наглядно показує скільки процесу потрібно часу для роботи з вказаний ресурсом.

Щоб оцінити коли ж все таки потрібно запуск перевірки, то це також залежить від конкретної операційної системи, дуже важливі фактори того, як часто виникають взаємні блокування в цій системі, які саме ресурси використовуються в системі та інше. Але більш менш впевнено можна запускати перевірку після того, як ресурс утримується процесом більше ніж вдвічі від середнього часу використання цього ресурсу процесами.

Також важливо відзначити, якщо використовувати лише вищевказаний спосіб, то це навряд чи відчутно підвищить швидкість реагування на ситуації взаємного блокування [7]. Тому для цього потрібно ввести ще один спосіб, який наблизить шанс більш швидкого виявлення взаємного блокування, цей спосіб зображено у формулі 2.

$$N = \frac{N_k * 100}{N_j} \quad (2)$$

У вищевказаній формулі зображено частоту запитів на використання щодо кожного ресурсу від всіх процесів відносно всіх задіяних в операційній системі ресурсів. У формулі N — коефіцієнт частоти запитів до ресурсу відносно інших ресурсів, N_k — кількість запитів за певну одиницю часу (яка визначається в кожній операційній системі індивідуально) до ресурсу, N_j — кількість всіх запитів за певну одиницю часу в операційній системі.

Суть формули 2 полягає в тому, що вона виводить частоту запитів кожного ресурсу в операційній системі за останній час її роботи. Як відомо, чим більше запитуваний процесами в системі ресурс, тим більша вірогідність того, що саме він і стане причиною взаємного блокування між процесами, яким він потрібен у подальшій для них роботі. Час за який потрібно оцінювати запитування кожного ресурсу процесами, також потрібно визначати від типу операційної системи та від того наскільки система в цілому навантажена в даний час. Плюсом цієї формули є те, що алгоритм розроблений на її базі також не буде впливати на роботу операційної системи і буде працювати незалежно від роботи процесів доки не виникне підозра у взаємному блокуванні.

Описані вище дві формули можна застосувати разом в роботі одного алгоритму, вони будуть значно підвищувати точність виявлення взаємного блокування і тим самим підвищать швидкість знаходження взаємного блокування в операційній системі. Коли ресурс утримується занадто довго, то запускається перевірка алгоритмом, що базується на другій формулі, якщо вона вказує на значну частоту запитів цього ресурсу, то це підтверджує підозри щодо виникнення ситуації взаємного блокування і дає привід для того щоб почати перевірку на наявність такого стану.

Запропонований спосіб звичайно не надасть можливості оперативно і якомога найшвидше реагувати на виникнення взаємних блокувань, але він підвищить точність їх виявлення в системах. Звичайно однією із проблем такого метода є те, що його потрібно для кожної операційної системи розробляти індивідуально, що ускладнює його застосування на практиці, але якщо все ж таки спробувати вирішувати проблему взаємних блокувань за допомогою такого метода, то відпаде необхідність у циклічній перевірці через кожний визначений в системі проміжок часу на наявність взаємних блокувань. Запропоновані методи на основі описаних формул базуються на задіяних ресурсів в операційній системі, хоча фактичною причиною виникнення взаємних блокувань в операційній системі є конкуренція між процесами та непередбачуваність їх дій від моменту створення. Тому можна вивести формули базуючись на природі роботи процесів в операційній системі. Хоча це буде значно складніше через декілька факторів [10].

По-перше, на відміну від ресурсів, які проблематично відібрати у процесу, який, в свою чергу, ним володіє, процеси динамічно створюються і знищуються під час роботи операційної системи. Це значно ускладнює розробку програми, що базується на основі запропонованого алгоритма, тому що потрібно створювати додаткові функції для відстеження процесів і фіксацію їх станів, створення нових процесів та знищення тих, що вже виконали свою роботу.

По-друге, в кожному процесу є свій пріоритет, який обов'язково потрібно врахувати, тому що від пріоритету залежить чи буде надано доступ певному процесу до запитуваного ним ресурсу, чи ні. Також параметр пріоритету визначає для алгоритму планування в системі скільки процесу потрібно виділити процесорного часу для обробки даних.

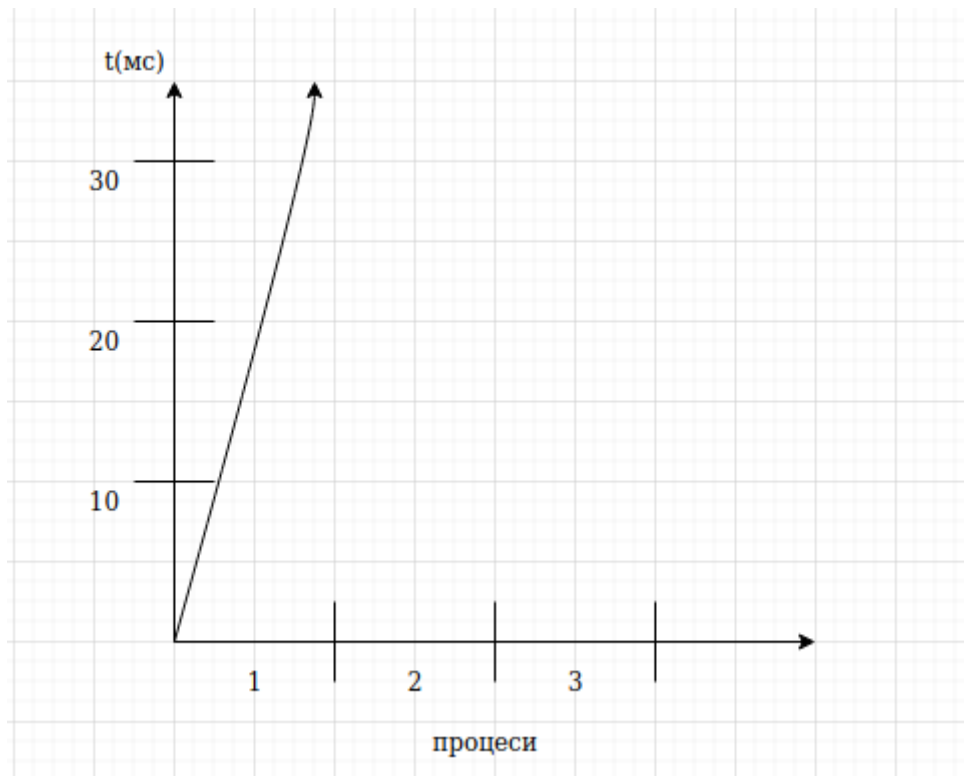


Рис. 5. Часова діаграма ресурсу, що довго утримується процесом

Дуже важливим фактором, який обов'язково потрібно враховувати, є час. Оскільки саме завдяки часу можна побачити те, наскільки висока вірогідність того, чи виникла ситуація взаємного блокування чи ні. Графік на рисунку 5 показує те, що ймовірність того, що виникла ситуація взаємного блокування значно висока. Вертикальна шкала графіка зображує час у мілісекундах, а горизонтальна - процеси, що подають запит на використання одного ресурсу. На вертикальній шкалі для кожного з процесів, що подають запит на використання ресурсу, виділено цілий відрізок.

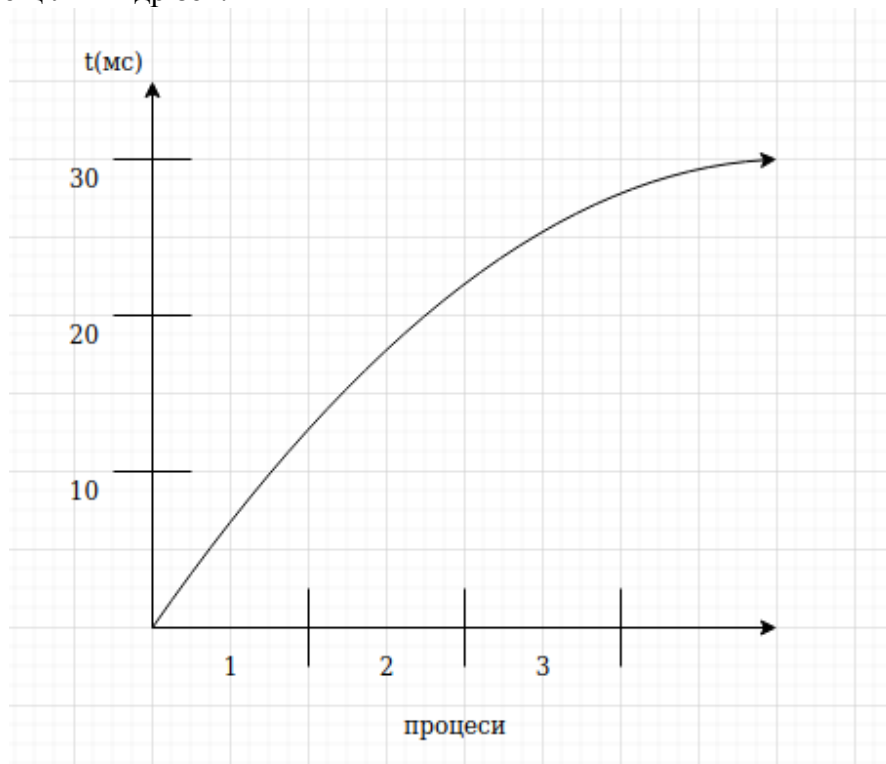


Рис. 6. Часова діаграма ресурсу, що надовго не утримується процесами

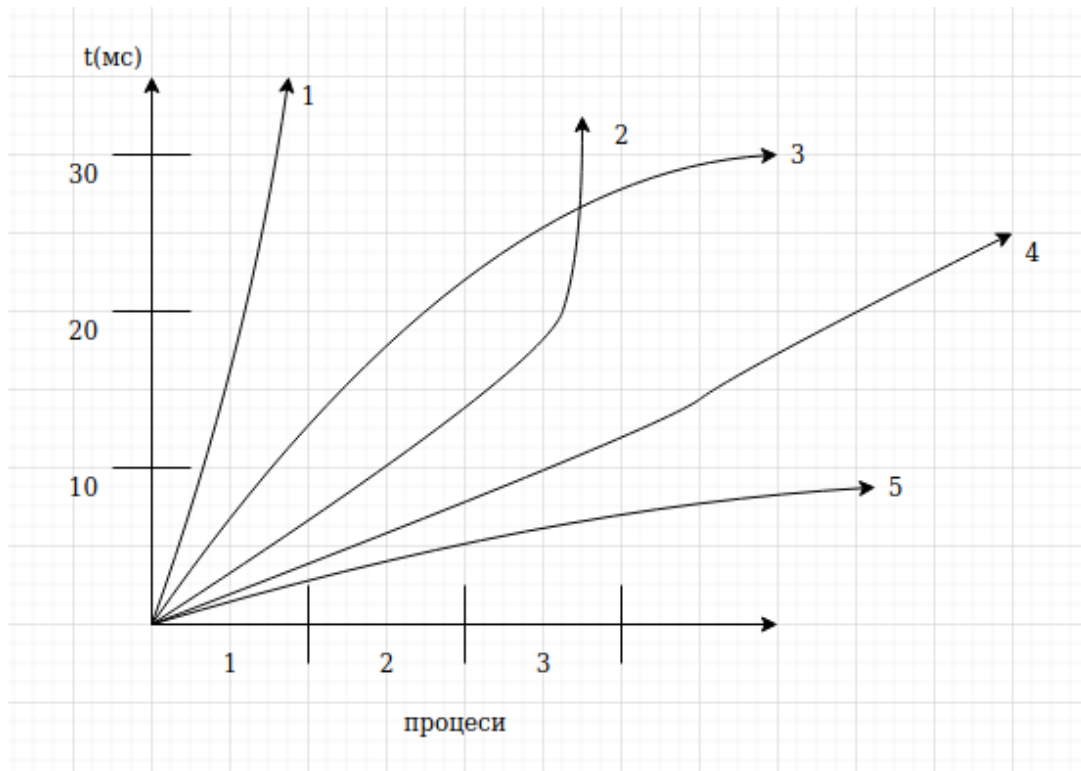


Рис. 7. Часова діаграма декількох ресурсів

Крива, яка описує ресурс, зображає те, який саме процес її утримує, в тому відрізку в якому знаходиться крива, той процес і утримує в своєму розпорядженні даний ресурс. На рисунку 5 явно видно, що перший процес занадто довго утримує в своєму розпорядженні ресурс, а другий і третій чекають на цей ресурс в черзі, отже висока вірогідність того, що виникло взаємне блокування.

На рисунку 6 зображена зворотна ситуація, коли процеси надовго не утримують в своєму розпорядженні ресурс та звільняють його після того, як виконали поставлені перед ними задачі операційною системою. Тому даний графік демонструє те, що вірогідність того, що виникла ситуація взаємного блокування значно мала.

Зображені та описані часові діаграми описують вірогідність виникнення ситуація взаємного блокування в операційній системі відносно ресурсу, при чому лише одного ресурсу [11]. Це зроблено для наочності, звісно можна в такій діаграмі зобразити дві і більше кривих, що описують по ресурсу і в залежності від того, які саме криві надовго затримуються в відрізках певних процесів, то ті криві і ті процеси і опинилися в заблокованому стані між собою, так як це зображено на рисунку 7.

Також такі самі графіки можна побудувати і для процесів, але зазвичай на графіку кривих буде більше, тому що кожна крива буде вказувати на запит для кожного ресурсу, що вже замість процесів зображені на горизонтальній шкалі. Вертикальна шкала так само відображає час утримання кожного з ресурсів процесом, який зображено на графіку. На рисунку 8 зображено процес, який утримував перший та другий ресурси, запити на використання ресурсів починаються з початку відрізка кожного ресурсу. Як видно з графіка, немає ризику виникнення ситуації взаємного блокування з цим процесом, тому що процес звільнив всі утримувані ним ресурси.

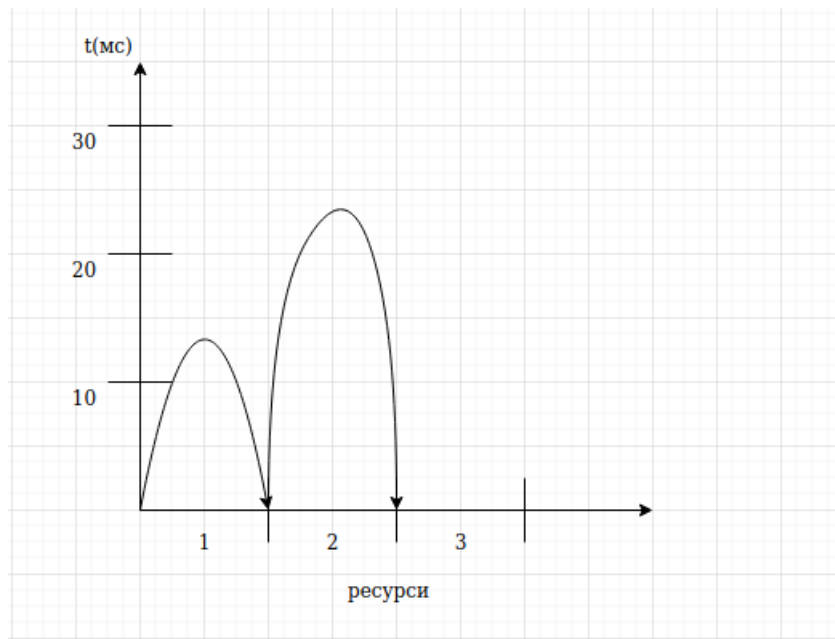


Рис. 8. Часова діаграма процесу

Графік, що зображено на рисунку 9 відображає високу вірогідність того, що виникла ситуація взаємного блокування, тому що запит процесу на другий ресурс дуже довго утримує даний ресурс. Як видно з самого графіка крива, що йде від запиту другого ресурсу, на відміну кривої, що зображує запит першого ресурсу, демонструє те, що даний процес навіть найближчим часом не збирається звільняти другий ресурс.

Наявність зображених графіків, ще раз підтверджують те, що набагато зручніше виявляти взаємні блокування слідкуючи саме за ресурсами, що були задіяні під час роботи операційної системи. Причиною цього являється те, що для опису процесів, потрібно значно більше кривих, які характеризують кожний запит від кожного процесу, тоді як для ресурсів просто можна зображати криві, що показують які саме процеси утримують даний ресурс в конкретний момент часу. Також факт того, що процеси, на відміну від ресурсів [12] (тих ресурсів, що не можна відібрати у процесу, не порушивши роботу системи), динамічно створюються під час роботи системи, що змушує створювати додаткові алгоритми для відстеження тих процесів, що вже були знищені, та тих, що були створені системою.

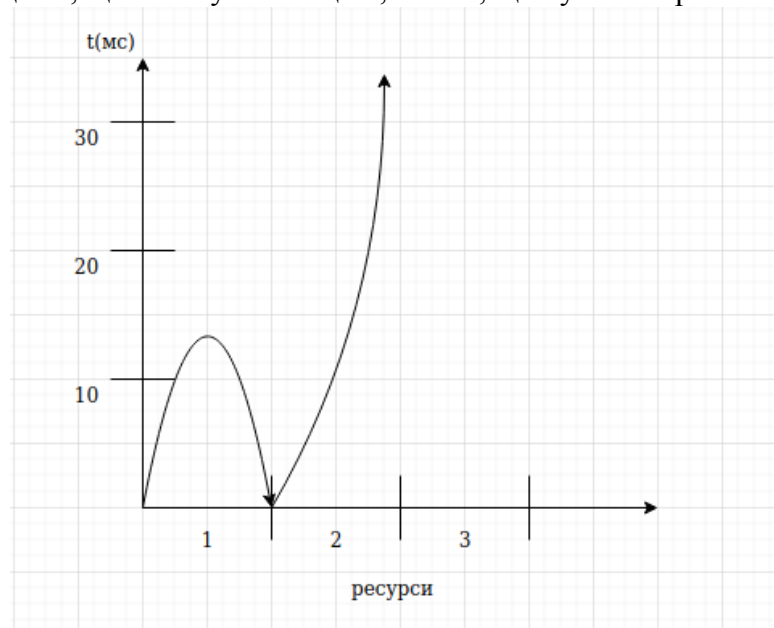


Рис. 9. Часова діаграма процесу у взаємному блокуванні

Висновки: В даній статті розглянута проблема виникнення взаємних блокувань в операційних системах. Розглянуто суть проблеми та причини її виникнення. Описані способи та методи вирішення даної проблеми шляхом її уникнення, також розглянуто алгоритми усунення взаємних блокувань. Описана проблема швидкого реагування на потенційне взаємне блокування та способи підвищення точності їх виявлення. Виведено формули на основі роботи процесів та алгоритму планування в операційних системах. Розглянуто способи вирішення вказаної проблеми, що ґрунтуються на спостереженнях за ресурсами. Також розглянуто варіант опису роботи процесів, що є більш складним у реалізації.

СПИСОК ЛІТЕРАТУРИ

1. Таненбаум Е. Сучасні операційні системи 4 видання. – СПб.: Пітер, 2015, - 1120с.
2. Лав Р. Ядро Linux. Описание процесса разработки 3 издание. - ООО Вильямс, 2013, 496с.
3. Стивенс У. Unix: Взаимодействие процессов — СПб.: Питер, 2003, - 576с.
4. Лав Р. Linux системное программирование 2 издание. - СПб.: Питер, 2014, - 448с.
5. Гордеев А.В. Системное программное обеспечение — СПб.: Питер, 2001, - 736 с.
6. Квиттнер П. Задачи, программы, вычисления, результаты. — : Мир, 1980. — 422 с.
7. Єфименко В.В., Оніщенко С.М., Франчук В.М. Операційні системи. Лабораторний практикум: Навчальний посібник. – К.: НПУ імені М.П. Драгоманова, 2008. – 124 с.
8. Ричард Петерсен. LINUX: руководство по операционной системе. Второе издание, переработанное и дополненное в двух томах, перевод. – Киев: BHV, 1998, 1000 с.
9. Шеховцов В.А. Операційні системи. – К.: Видавнича група ВНУ, 2005. – 576 с.
10. Котов В. Е. Сети Петри. — М.: Наука, 1984. — 160 с.
11. Питерсон Дж. Теория сетей Петри и моделирование систем. — М: Мир, 1984. — 264 с.
12. Андрей Лобачевский. Операционная система UNIX. BHV-СанктПетербург. 1997, 500 с.

ЗАСТОСУВАННЯ ДВІЙКОВОГО КОДУВАННЯ РОЗРІЗІВ ДЛЯ ВИРІШЕННЯ МЕРЕЖНИХ ЗАДАЧ

Князева Н.О., Лисенко Н.О.

Анотація. В роботі показано, що завдання визначення пропускної здатності, надійності та живучості набувають все більшої актуальності як в сфері проектування, так і при експлуатації телекомунікаційних мереж зважаючи на інтенсивний розвиток телекомунікацій в останнє десятиріччя.

Запропоновано методи вирішення мережних задач, що базуються на застосуванні двійкового кодування розрізів мережі, а саме: визначення максимального двохполюсного потоку одночасно для всіх пар пунктів мережі; визначення коду мінімального розрізу; визначення нижньої межі структурної надійності та структурної живучості телекомунікаційних мереж; визначення максимального двохполюсного потоку на основі застосування множини розрізів у диз'юнктивній нормальній формі.

Використання двійкового кодування розрізів призводить до чіткої формалізації процесів вирішення вказаних задач та в результаті – до практичної можливості застосування запропонованих методів для сучасних телекомунікаційних мереж.

Annotation. It is shown, that task of determination of carrying capacity, reliability and vitality are acquired all greater actuality both in the sphere of planning and during exploitation of telecommunication networks having regard to intensive development of telecommunications in the last decade. The methods of decision of network tasks that are based on application of binary code