FRI-2G.303-1-CCT1-07

TEACHING OPERATING SYSTEMS: BANKER'S ALGORITHM [7]

Tzanko Golemanov, PhD
Department of Computer Systems and Technologies,
"Angel Kanchev" University of Ruse
Tel.: +359 82 888 681
E-mail: TGolemanov@uni-ruse.bg

Emilia Golemanova, PhD
Department of Computer Systems and Technologies,
"Angel Kanchev" University of Ruse
Tel.: +359 82 888 681
E-mail: EGolemanova@uni-ruse.bg

Abstract: *Studying operating systems helps in understanding computer architecture, how different components of a computer interact, and how system resources are utilized. Deadlock is a potential risk for any computing system, where it most often occurs when resources are allocated. The main topics in Deadlock teaching are: Definition and Characteristics of Deadlock, Necessary Conditions for Deadlock, Prevention Techniques for Deadlock, Deadlock Avoidance Techniques, and Deadlock Detection and Recovery. The Banker's algorithm (developed by Edsger Dijkstra) is a resource allocation and deadlock avoidance algorithm that is used in computer operating systems. In this paper, we would like to share our experience in teaching Banker's algorithm using a specially developed tool BANKER. The tool allows students to learn in detail the steps of the algorithm, its capabilities, and its limitations, as well as to experiment with a different number of processes and resources.*

Keywords: *Operating Systems, Teaching Tools, Deadlocks, Banker's algorithm*

ВЪВЕДЕНИЕ

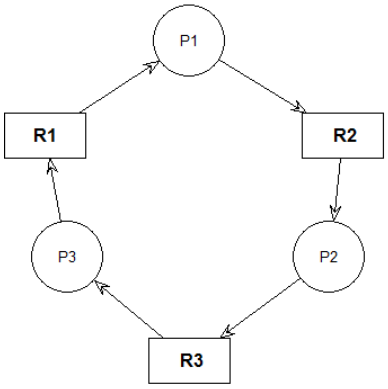
Deadlock (*Взаимна блокировка, Клинч, Тупик*) както е известно, е ситуация без изход. Въпреки че за първи път на този проблем се отделя сериозно внимание в теорията и практиката на Операционните системи (ОС) (Tanenbaum and Bos, 2016) (Silberschatz, Galvin and Gagne, 2013), (Deitel, Deitel and Choffnes, 2004), такива ситуации са често срещано явление и в реалния живот, особено в области, където трябва да се разпределят и използват ограничени по количество ресурси. Един процес в Изчислителната система (ИС) се намира в състояние Deadlock, ако очаква настъпването на събитие, което никога няма да настъпи. Взаимното блокирането е изключително сериозен проблем и представлява потенциална опасност за всяка компютърна система, когато много процеси (понякога стотици) остават блокирани за неопределено (потенциално безкрайно) време. Алгоритъмът на Банкера се реализира в модул на ОС, който осигурява висока степен на контрол при разпределение на ресурсите на ИС. По този начин се цели предотвратяване на блокирането й и се гарантира безопасната й работа.

Deadlock е една от концепциите в областта на ОС, които често са трудни за усвояване от студентите. Затова тук бихме искали да споделим нашия опит в преподаването на този материал в Русенския университет, като акцентът е Алгоритъмът на Банкера, особеностите при реализацията му по време на практическите упражнения, както и основната функционалност на авторския софтуерен инструмент за преподаването му Banker.exe.

DEADLOCK: Примери, необходими условия, направления в изследванията

[7] Докладът е представен на научната сесия на 27.10.2023 в секция „Комуникационна и компютърна техника" с оригинално заглавие на български език: ОБУЧЕНИЕ ПО ОПЕРАЦИОННИ СИСТЕМИ: АЛГОРИТЪМ НА БАНКЕРА.

По време на лекциите на студентите се обясняват базовите принципи, свързани с понятието Deadlock, необходимите условия за възникването му и основните направления, в които се провеждат изследвания за справяне с проблема. В една ИС Deadlock възниква най-често в резултат на конкуренцията за ресурси, изискващи последователно използване от процесите, т.е. когато процесите трябва да имат монополни права за достъп да тези ресурси. За визуално онагледяване на възникването на проблема се използва графа за разпределяне на ресурсите в ИС, при означенията показани на Фиг.1:



Deadlock: Характеризира се с наличието на циклична верига от блокирани процеси, всеки от които задържа ресурс, който е необходим на предходния от веригата: Процес P1 задържа ресурс R1, като същевременно изисква ресурс R2, който е зает от процес P2.

Фиг.1. Графично представяне на Deadlock

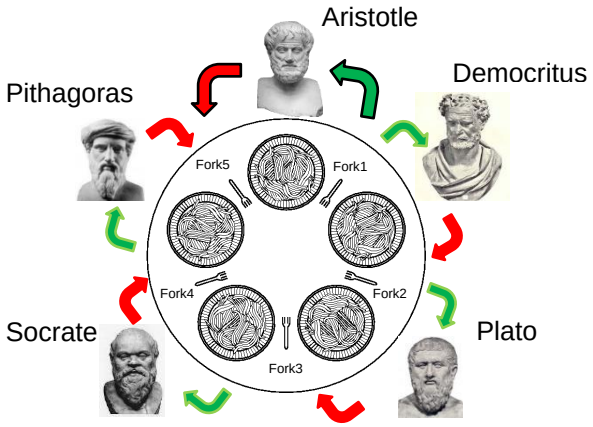
За първи път на проблемът Deadlock се отделя сериозно внимание в класическата статия (Coffman, Edward G., Elphick and Shoshani, 1971). Според изследователите на проблема взаимна блокировка може да възникне, когато в системата са изпълнени едновременно следните **необходими условия** (1):

1. ***Взаимно изключване***: Процесите изискват **монополни права** на използване на предоставените им ресурси - само един процес може да използва ресурса за определен период от време.
2. ***Заявка за нов ресурс***: Процесите задържат предоставените им ресурси, като могат по всяко време да поискат от системата **нов допълнителен ресурс** и чакат, ако е зает.
3. ***Непреразпределяемост***: Процесите не освобождават ресурсите, **докато не завършат** докрай работата си с тях, т.е. ресурсите не могат да бъдат преразпределяни, като се отнемат от едни процеси и се предоставят на други. (1)
4. ***Кръгово очакване:*** Допустимо е възникването на циклична верига от процеси, в която всеки процес задържа ресурс, необходим на предходния процес от веригата.

Търсенето на решение за справяне с Deadlock е в следните направления:

[1] **Предотвратяване на възможността** за възникване на Deadlock в системата;
[2] **Динамично избягване** на Deadlock чрез внимателно разпределение на ресурсите;
[3] **Локализиране** на Deadlock в системата;
[4] **Възстановяване** на системата след Deadlock.

Ясно е, че най-коректният подход за справяне с Deadlock е в направление [1], където се цели получаване на система, в която възникването на взаимна блокировка да е принципно невъзможно. Това се постига при нарушаване на поне едно от необходимите условия в (1). До такова решение (Фиг.2) студентите стигат при решаване на класическата задача за Обядващите философи (Dijkstra, 1971), където програмно може да се зададе заявките за нови ресурси (вилици) да бъдат винаги във възходящ ред на номерата. По този начин се създава система, в която не е възможно възникването на Кръгово очакване.

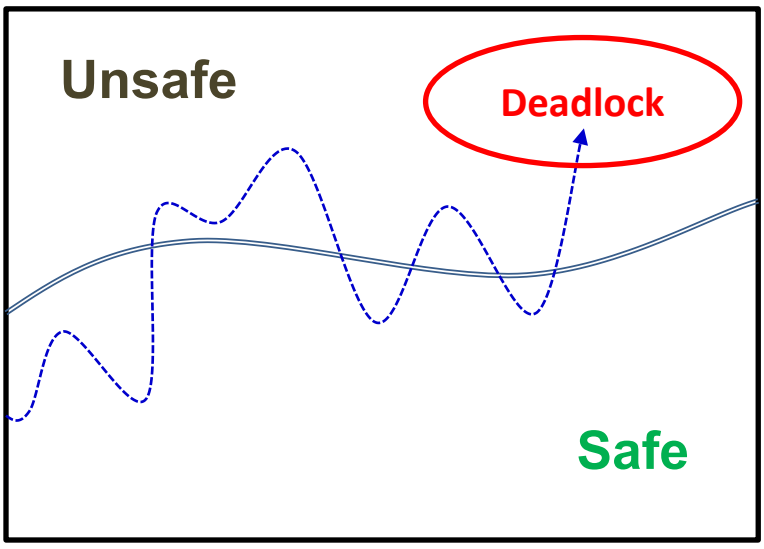



```
void Aristotle ()
semaphore Fork1, Fork5;
{
while ( true )
 {
   cout << "Think" << #13;
    wait ( Fork1 );
    wait ( Fork5 );
     cout << "Eat" << #13;
    signal ( Fork1 );
    signal ( Fork5 );
  }
}
```

Фиг.2. Предотвратяване на възможността за възникване на Deadlock чрез нарушаване на условие 4. Кръгово очакване

Алгоритъм на Банкера за избягване на DEADLOCK

Избягване на Deadlock (Deadlock Avoidance) - в този случай Deadlock потенциално би могъл да възникне, но ОС следи непрекъснато състоянието на системата и ресурсите и при увеличаване на вероятността за поява на Deadlock се предприемат съответни мерки за избягването му. Най-известен в това направление е Алгоритъмът на Банкера, предложен от Dijkstra, базиран на концепцията му за Сигурни и Несигурни състояния на системата.



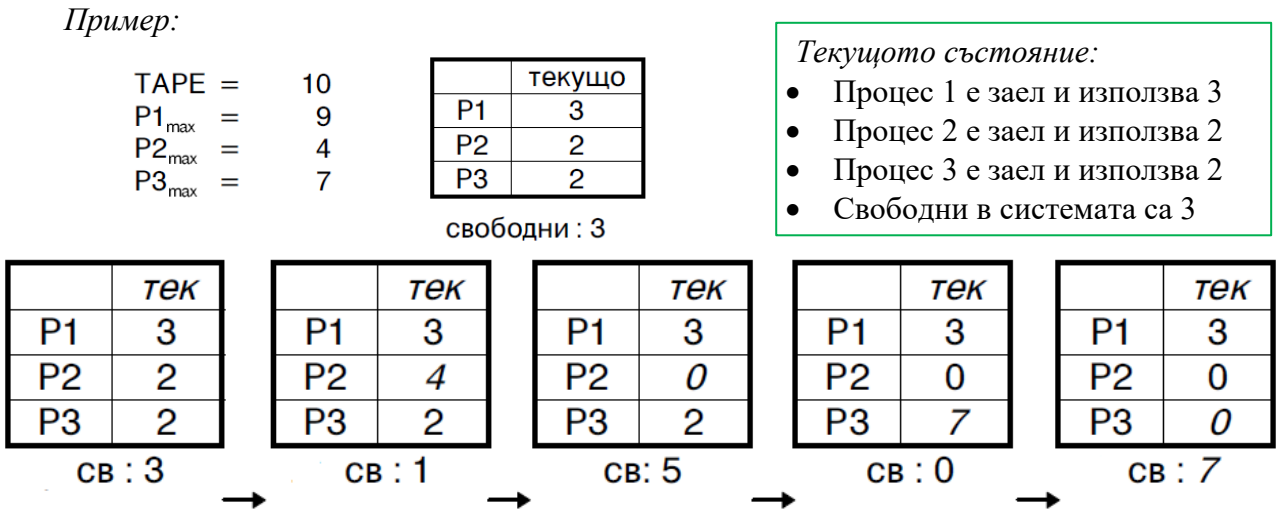
Фиг.3. Динамика на използването на ресурсите от процесите

На Фиг.3 е показана принципно динамиката на използването на ресурсите от процесите и преминаването на системата в различни състояния:

- **Сигурно (Safe)** състояние: Гарантирано е удовлетворяването на всякакви последователности от бъдещи заявки за ресурси. Допустимо е забавяне на изпълнението на някои заявки дори при наличие на ресурси!
- **Несигурно (Unsafe)** състояние: Някои последователности от заявки за ресурси могат да доведат до Deadlock.
- **Обречено състояние**: Всички възможни действия водят до Deadlock.

Идеята на избягването на Deadlock е ОС така да управлява и разпределя ресурсите, че да не допуска никога навлизане в Несигурната зона. За целта се изисква от всички процеси предварително да обявят пиковите си заявки за ресурси (при което успешно завършват), а

ОС се грижи текущото състояние да е винаги **Сигурно**, т.е. да има **поне една последователност** от раздаване на ресурси, при която се удовлетворяват максималните заявки на **всички процеси.** На Фиг.4 е показан илюстративен пример за едно текущо състояние на система при три процеса с максималните им заяки и проверката дали това състояние е Сигурно (Safe).



Фиг.4. Проверка за Сигурно състояние

Интерфейс и функционалност на симулатора Banker

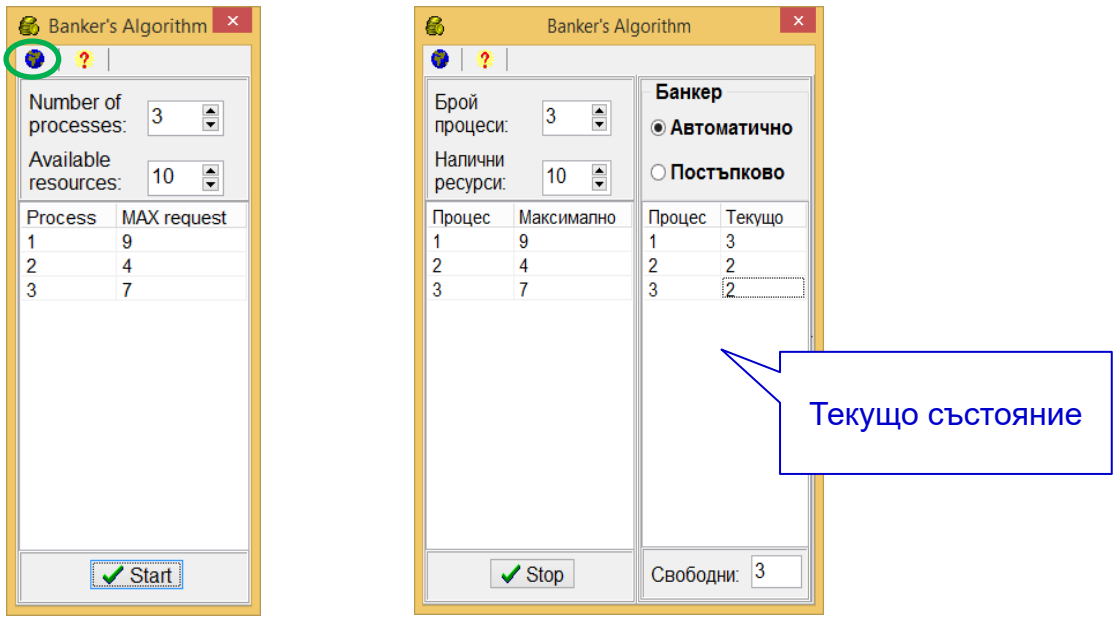
Софтуерният симулатор Banker е авторски продукт, предназначен за обучение по Deadlock в курса „Операционни системи“ на Русенския университет. Стремежът при разработката му е да бъдат избягнати редица недостатъци и/или ограничения на налични подобни обучаващи системи:

- Реализации като WEB приложения – по принцип с облекчен достъп, но и по-неудобен и ограничаващ потребителски интерфейс;
- Неоправдано усложнена функционалност, изискваща продължителен период на обучение за използването;
- Предварително фиксиран брой на процесите и ресурсите, с които се извършва експериментирането;
- Липса на многоезичен потребителски интерфейс.

При отчитане на горните съображения, основните характеристики на представяната разработка са:

- Desktop базирана софтуерна система, без необходимост от инсталационна процедура;
- Опростен потребителски многоезиков интерфейс, разширяващ се при преминаване към нова функционалност;
- Гъвкаво задаване на броя процеси и ресурси при решаване на всяка конкретна задача.

На Фиг.5 е показан началният вид на симулатора с маркирана възможността за промяна на езика на интерфейсните текстове.

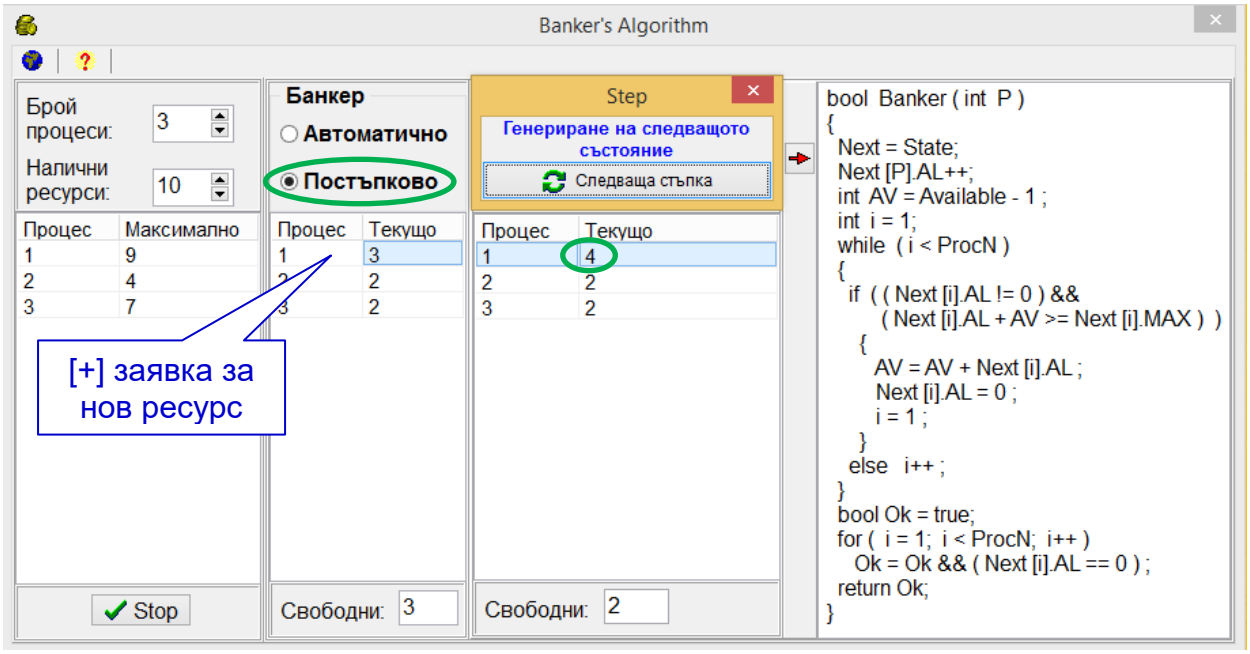


(a) (b)

Фиг.5. Стартиране на Banker (a) и формиране на текущо състояние на системата (b)

Симулаторът демонстрира действието на алгоритъма на Банкера в два основни режима - Автоматично и Постъпково. При всяка заявка на процес за нов ресурс в Автоматичен режим резултатът е или удовлетворяване на заявката или съобщение за отказ (в случай, че системата ще премине в Несигурно състояние).

При избор на режим на Постъпково изпълнение интерфейсът на системата се променя във вида, показан на Фиг.6.



Фиг.6. Постъпково изпълнение на Алгоритъма на Банкера

Добавят се нови области на приложението, съдържащи вътрешните структури на следващото състояние на системата и прозорец със C-style псевдокод на Алгоритъма на Банкера.

На фигурата е показано как при заявка на процес 1 за един нов ресурс се генерира следващото състояние (с увеличен брой заети ресурси) и се проверява дали то е Сигурно. Изпълнението на тази проверка се визуализира постъпково чрез движението на стрелката-указател отстрани на псевдокода на алгоритъма и коментари за действията, извършвани на всяка стъпка. Паралелно се следят и промените в съдържанието на вътрешните структури на състоянието на системата.

ИЗВОДИ

Изучаването на концепциите и базовите принципи на изграждане и функциониране на модули на операционните системи представлява сериозно предизвикателство за студентите по компютърни системи и технологии. В тази материал са представени възможностите на авторския инструмент Banker, специално проектиран да се използва в дисциплините Операционни системи. Демонстрира се прилагането на Banker като ефективно средство при изучаването на избягването на Взаимна блокировка (Deadlock) при конкурентни паралелни процеси. Възможността за интерактивно задаване на процеси и използваните от тях ресурси, както и наблюдението на постъпковото изпълнение на алгоритъма на Банкера, позволява на студентите да се ориентират по-добре в преподавания теоретичен материал.

Banker се използва от няколко години в катедра „Компютърни системи и технологии“ на Русенския университет и отзивите на студентите са положителни. Липсата на каквато и да е инсталационна процедура, малкият размер и copyleft-лицензът позволяват Banker да бъде използван без ограничения в произволна MS-Windows среда.

ACKNOWLEDGEMENTS

Публикацията е с подкрепата на проект 23-ФЕЕА-01 „Разработване на модели и симулации с различни области на приложение“, финансиран от фонд „Научни изследвания“ на Русенския университет „Ангел Кънчев“.

REFERENCES

Coffman, Edward G., J., Elphick, M. J. and Shoshani, A. (1971) ‘System Deadlocks’, ACM Computing Surveys. doi: 10.1145/356586.356588.

Deitel, H. M., Deitel, P. J. and Choffnes, D. R. (2004) Operating systems. Pearson/Prentice Hall.

Dijkstra, E. W. (1971) ‘Hierarchical ordering of sequential processes’, Acta Informatica, 1(2), pp. 115–138.

Silberschatz, A., Galvin, P. B. and Gagne, G. (2013) Operating System Concepts. 9. John Wiley & Sons, Inc.

Tanenbaum, A. S. and Bos, H. (2016) Modern Operating Systems, Education. Pearson India; 4th edition.