FRI-ONLINE-1-CCT1-04

TEACHING OPERATING SYSTEMS: DEADLOCKS[4]

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: *Process Synchronization and Deadlocks are some of the toughest topics of Operating Systems and Computer Science in general. It is necessary to solve a lot of exemplary tasks just to strengthen the understanding of the concepts but they still remain difficult to grasp by the students. A set of highly interactive, integrated tools has been designed specifically to support both teaching and learning Operating Systems at the University of Ruse. This paper presents a brief description of the Deadlock problem as well as the features of the basic teaching tool used in the Operating Systems course. This tool allows the students to develop and run concurrent programs in two simple programming languages and to monitor the information in the main system tables. A concise and relatively self-contained description of how the tool has been used to help students understand the concept of Deadlock and in particular Deadlock Prevention is given.*

Keywords: *Operating Systems, Teaching Tools, Deadlocks, Concurrent Programming.*

ВЪВЕДЕНИЕ

Deadlock (*Взаимна блокировка, Клинч, Тупик*), е една от концепциите в областта на операционните системи (ОС), които често са трудни за усвояване от студентите и затова тук бихме искали да споделим нашия опит в преподаването на този материал в Русенския университет, като акцентът е в работата по време на практическите упражнения.

Deadlock както е известно, е ситуация без изход. Въпреки че за първи път на този проблем се отделя сериозно внимание в теорията и практиката на 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. 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. В този смисъл появата на Deadlock-ситуация в системите, контролиращи важни и животозастрашаващи дейности, е абсолютно неприемливо, а дори обсъждането на евентуалните й прояви трябва да е табу. Следователно изучаването на проблема Deadlock в курса на ОС, както и начините за справяне с него е изключително важно за студентите по компютърни системи и технологии.

Курсът по операционни системи в бакалавърската степен може да бъде сериозно предизвикателство за всеки преподавател по компютърни науки. Студентите (много от които нямат почти никакъв опит в системното програмиране) трябва да се запознаят с теоретичен материал относно принципите на изграждане на на модулите на една ОС, както и механизмите на тяхното прилагане. Както преподавателят, така и студентите се сблъскват с много трудности, особено по време на практическите упражнения. Една комерсиална операционната

[4] Докладът е представен на заседание на секция 3.2 на 29 октомври 2021 с оригинално заглавие TEACHING OPERATING SYSTEMS: DEADLOCKS

система не е съвсем подходяща за целите на обучението поради необходимостта студентите да експериментират като модифицират в движение основните й настройки и следят отблизо резултатите от тези промени върху стартираните процеси. Тук преподавателят се нуждае от удобен симулационен софтуер, чрез който да обяснява на практика различни концепции на ОС. Този софтуер за обучение не трябва да бъде твърде сложен и труден за използване (целта на курса на ОС не е да овладее работата с някакъв симулатор), но същевременно да бъде достатъчно мощен, визуален и удобен за използване, за да бъде от реална помощ на студентите. За съжаление, повечето от използваните симулационни пакети попадат в една от двете крайности - някои (MINIX, Nachos, Pintos, GeekOS, YASS) са наистина мощни, но много сложни за усвояване и използване. Други (OS Algorithm Simulator, MOSS Deadlock Simulator) са с твърде специализирани, с ограничена фунционалност, недостатъчно визуални и неатрактивни. Идеята за използване на версии на такива симулатори за мобилни устройства само влошава положението - ограниченото екранно пространство води до необходимостта от преминаване през множество екрани и менюта, дори и за съвсем прости настройки или действия.

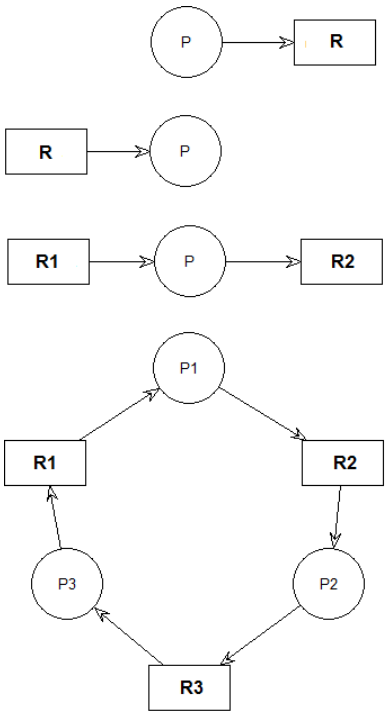
Използването на единствен симулатор за целия курс на ОС е идея, която е много амбициозна и трудна за реализация. Някои теоретични концепции са толкова сложни, че един симулатор не е достатъчен. Такава концепция е Deadlock и за да изясним детайлите в ние използваме три различни подхода и три различни вида софтуер за симулация. Всеки от тях се фокусира върху различен аспект на Deadlock и позволява на студентите сами и по визуален начин да изследват проблемите и да намерят правилни решения. В настоящия доклад се представя основния обучаващ софтуерен модул, позволяващ да се симулира Deadlock и да се търси решение в посока предотвратяване на появата му в системата.

Структурата на доклада продължава с: Изложение, в което е направен кратък Преглед на проблема Deadlock. Коментират се (без подробности) само тези аспекти на Deadlock, които ще бъдат обсъждат по-късно в обучаващия симулатор. След това е представена технологията на процеса на обучение. Дефинирани са основните стъпки през които преминават студентите по време на упражненията. Изложението продължава с описание на използването на обучаващия софтуер при преподаване на Deadlock, дефинирана е задачата за Обядващите философи (Dijkstra, 1971), проблемите и търсенето на решение. В Заключението е представена оценката на възможностите на обучаващия софтуер както и някои изводи за преподаването на Deadlock в бакалавърския курс на Русенския университет.

DEADLOCK: Преглед на проблема

Взаимна блокировка може да възникне, когато на процесите изискват монополни права за достъп до общите ресурси (хардуерни устройства или структури от данни), т.е. за определен период от време само един процес може да използва предоставения му ресурс. За системен Deadlock се говори, когато е налице циклична верига от процеси и ресурси, където всеки от процесите задържа ресурс, който е необходим на предходния във веригата. Изчакването за освобождаването на зает ресурс не е нещо необичайно за процесите по време на изпълнението им, но проблемът в този случай е, че това ще продължи до безкрайност. Стотици процеси могат да бъдат включени в такава циклична верига, което води до тотално блокиране на цялата система.

Графично представяне на Deadlock е показано на Фиг.1.



a) Процес P генерира заявка за използване на ресурс R

b) Ресурс R е предоставен за използване на процес P и той го задържа

c) Процес P задържа и използва ресурс R1, като същевременно генерира заявка за нов ресурс R2

d) **Deadlock**: Циклична верига от блокирани процеси, всеки от които задържа ресурс, който е необходим на предходния от веригата

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. ***Взаимно изключване***: Процесите имат изключителен контрол върху ресурсите - само един процес може да използва ресурс за определен период от време.
2. ***Заявка за нов ресурс***: Процесите задържат предоставените им ресурси и по всяко време могат да изискват нови и чакат ако са заети. (1)
3. ***Непреразпределяемост***: Ресурсите не могат да бъдат преразпределяни, като се отнемат от процеси, преди да завършат работата си с тях.
4. ***Кръгово изчакване***: Допустимо е възникването на кръгово изчакване от процеси в графа за разпределение на ресурсите.

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) Откриване на Deadlock и възстановяване на системата

Ясно е, че най-коректния път за справяне с Deadlock е в направление 1) където се цели получаване на система, в която възникването на взаимна блокировка е принципно невъзможно. Това се постига при нарушаване на поне едно от необходимите условия в (1).

DEADLOCK: Технология на учебния процес и учебен софтуер

Лекторът запознава студентите с теоретичните аспекти на проблема Deadlock, обяснява как възниква и представя някои класически примери, коментирани в литературата. За да се справят успешно с тази част на учебния материал, студентите трябва да разгледат проблема по-отблизо и да експериментират с причините за неговото възникване и подходите за разрешаване му.

Следователно основните етапи в преподаването на Deadlock на нашите студенти включват:

1. Запознаване с теоретични принципи и алгоритми, свързани с проблема Deadlock - по време на лекции;
2. Илюстриране на теоретичните принципи чрез практически примери - по време на практическите упражнения;
3. Самостоятелно проектиране и разработване от студентите на специфични алгоритми за разрешаване на проблеми, свързани с Deadlock.

Особено важно в тази последователност от дейности е използването на подходящ учебен софтуер, който:

1. Да илюстрира достатъчно ясно концепциите на Deadlock, обсъдени в лекциите;
2. Да помогне на студентите при разработката на софтуерните им проекти.

В същото време, според нас, използвания учебен инструментът трябва да отговаря на следните основни изисквания:

• да работи на достъпна хардуерна и софтуерна платформа;
• да поддържа на мултипрограмиране и паралелни процеси;
• да е с познат и незатрудняващ интерфейс;
• да поддържа познати езици за програмиране с възможност за паралелна обработка на информацията.

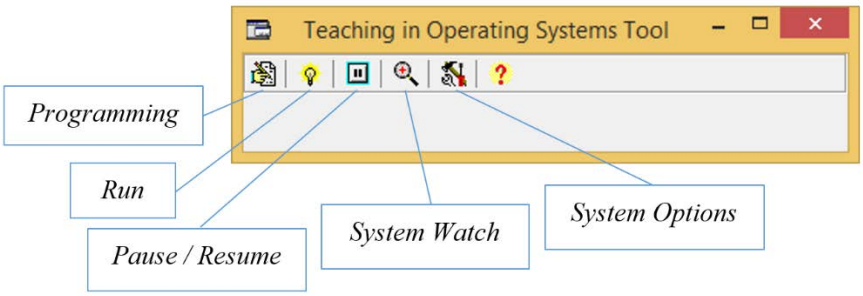
Интегрираната среда TOST (Teaching in Operating Systems Tool) е разработена точно с тази идея. Представлява виртуална машина (Golemanov and Golemanova, 2006), позволяваща на студентите да разработват и изпълняват паралелни процеси, като в същото време могат да променят основните настройки на операционната система (стратегия за планиране на процесорното време, размер на кванта в режим на времеделение, стратегия за управление на паметта, стратегия за управление на външната памет и т.н.), както и да наблюдават основните системни таблици (Процеси, Блокирани, Готови, Семафори и др.).

Разглежданата интегрирана среда включва:

• многозадачна и многопрозоречна операционна система
• редактори и компилатори за два прости езика за програмиране
• емулатор на виртуален процесор.

Текущата версия на TOST (2021) е разработена посредством на Embarcadero® Delphi в средата на Embarcadero RAD Studio 10.3.3 Rio. Изпълнимият файл е много малък по размер (по-малко от 1 MB), няма инсталационна процедура и може да се стартира във всяка MS Windows среда. TOST е лесна за усвояване и използване и позволява на студентите да се започнат пълноценна работа с нея за кратко време (около час).

Общият вид на главно меню на TOST е показан на Фиг.2.



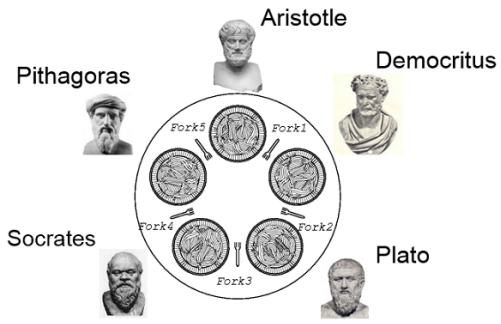
Фиг. 2. TOST интерфейс

За демонстрация и анализ на някои от сериозните проблеми при изпълнението на паралелни процеси, както и на подходите за търсене решение, често се използва задачата за Обядващите философи (Dining Philosophers).

DEADLOCK: Задача за Обядващите философи

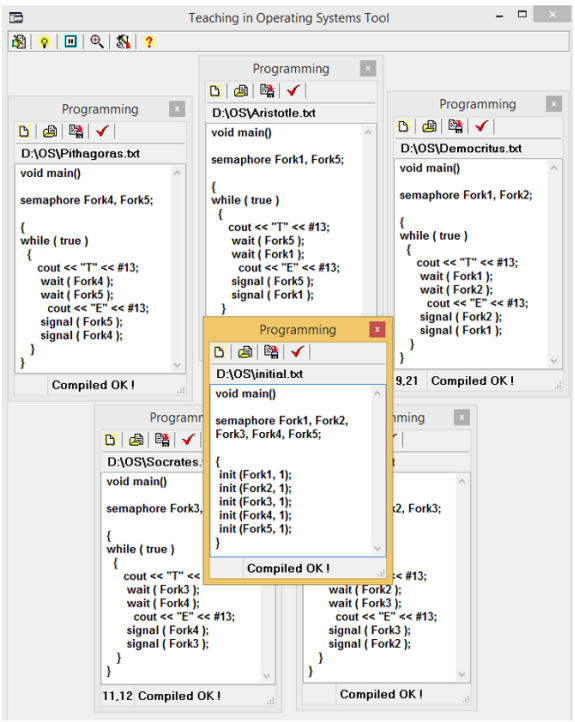
Дефиниция: Класическата задача (Dijkstra, 1971) има следната дефиниция:

Петима приятели философи имали обичай да се събират и да обядват докато обменят мисли, идеи и концепции. За целта използвали малък италиански ресторант, който разполагал само с една кръгла маса, пет чинии и пет вилици, разположени между всеки две чинии.

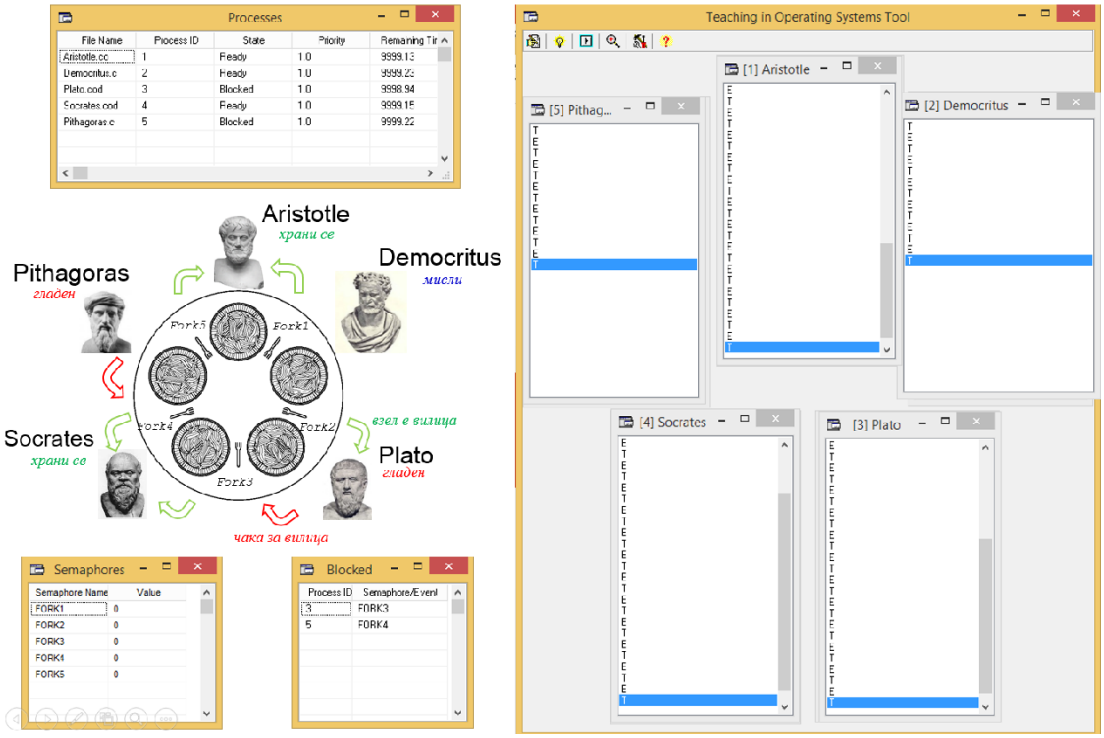


Жизненият цикъл на всеки от философите включва:

1. ***Мислене** и разговор с другите философи*
2. *Вземане една и след това и втората вилици*
3. ***Похапване** на спагети*
4. *Оставяне на една и втора вилици*
5. *Преход към т.1.*



Фиг. 3. Задача за Обядващите философи – дефиниция и програмиране



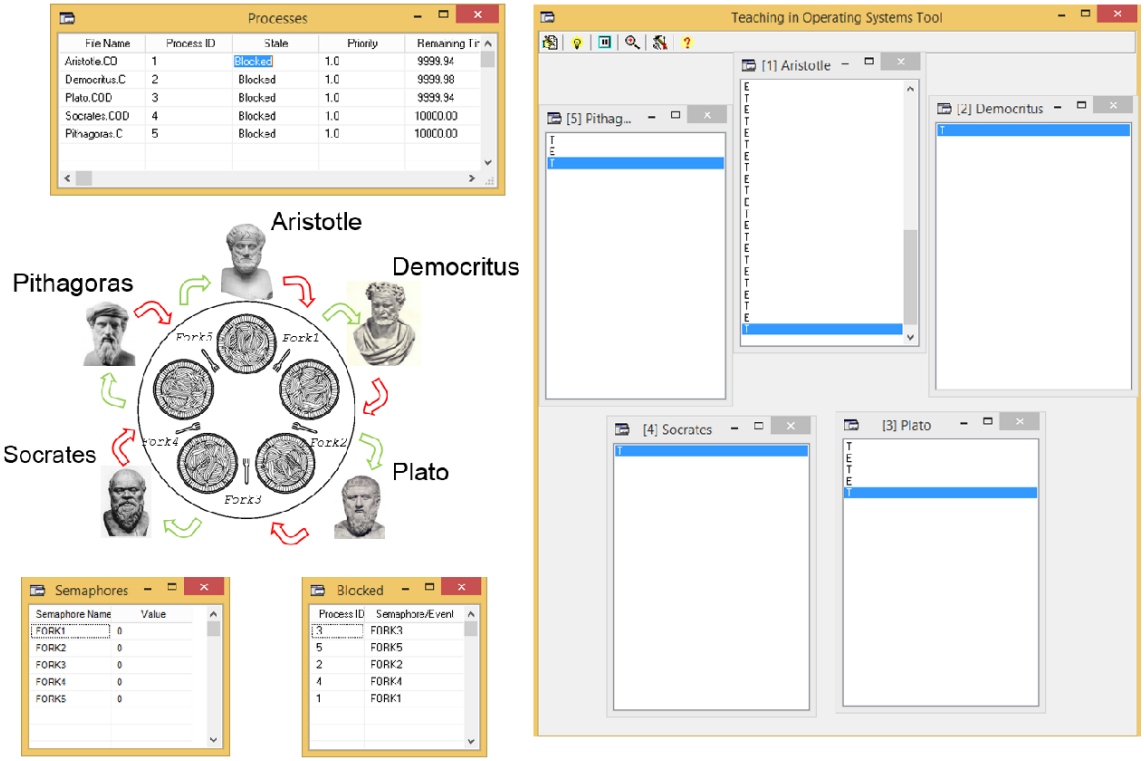
Фиг. 4. Задача за Обядващите философи – изпълнение на процесите

Разработка: Студентите решават самостоятелно задачата чрез разработване на пет програми за организация на жизнения цикъл на философите, като за реализация на коректен достъп до разделяемите ресурси (вилиците) се използват семафори. На Фиг.3 е показана

средата TOST и сорс-кодовете на програмите – по една за всеки от философите и още една инициализираща за задаване на началните стойности на използваните семафори.

Реализация: TOST дава възможност на студентите да наблюдават (Фиг.4), при реалното изпълнение на стартираните процеси, как философите чакат за вилици (в таблиците Processes и Blocked) и как вземат и връщат вилиците на масата (в Semaphores).

Демонстрация на Deadlock: В TOST лесно и нагледно се демонстрира и възникването на Deadlock в системата. След известен период на изпълнените на процесите се вижда как всеки един от философите взема вилица с дясната си ръка, и след това се блокира в очакване на освобождаването на втора вилица (Фиг.5).



Фиг. 5. Задача за Обядващите философи – **Deadlock**

Решение: Студентите трябва да намерят решение и да спасят философите от гладна смърт. Една от идеите е да се наруши четвъртото необходимо условие за Deadlock (1) – условието за ***Кръгово изчакване***.

```
void main()
semaphore FORK5, FORK1 ;
{
while ( true )
  {
     cout << "T" << #13;
    wait ( FORK5 );
    wait ( FORK1 );
      cout << "E" << #13;
    signal ( FORK1 );
    signal ( FORK5 );
  }
}
```

```
void main()
semaphore FORK5, FORK1 ;
{
while ( true )
  {
     cout << "T" << #13;
    wait ( FORK1 );
    wait ( FORK5 );
      cout << "E" << #13;
    signal ( FORK1 );
    signal ( FORK5 );
  }
}
```

Фиг. 6. Задача за Обядващите философи – програма на **Aristotle** - Преди и След промените

Студентите прилагат този подход, като препрограмират **Aristotle** (Фиг.6) и въвеждат общото изискване за всички процеси (философи) да заемат ресурси (вилици) **само във възходящ ред** на номерата им. Така, предотвратявайки възможността за затваряне на

цикличната верига (от Aristotle**)** се постига и предотвратяване на възможността за възникване на Deadlock**.**

ЗАКЛЮЧЕНИЕ

Изучаването на концепциите и базовите принципи на изграждане и функциониране на операционните системи представлява сериозно предизвикателство за студентите по компютърни системи и технологии. В тази материал са представени някои от възможностите на авторска интегрирана среда, наречена TOST, специално проектирана да се използва в дисциплините Операционни системи и Паралелно програмиране. Демонстрира се прилагането на TOST като ефективно средство при изучаването на Взаимна блокировка при конкурентни паралелни процеси. Възможността за наблюдение в реално време на информацията от основните таблици на операционната система позволява на студентите да се ориентират по-добре преподавания теоретичен материал.

TOST 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 TOST да бъде използван без ограничения в произволна MS-Windows среда.

ACKNOWLEDGEMENTS

Този доклад се публикува с подкрепата на проект 21-ФЕЕА-01 „Интелигентни компютърни системи: изследване на тяхното развитие, приложение и управление“, финансиран от фонд „Научни изследвания“ на Русенски 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.

Golemanov, T. and Golemanova, E. (2006) ‘A Teaching in Operating Systems Tool’, in *CompSysTech’06*.

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.