A Teaching in Operating Systems Tool

Tzanko Golemanov, Emilia Golemanova

Abstract: *The paper describes the main features of the integrated environment TOST. The objective of TOST is to be used in Operating Systems, Parallel Programming and Compilers courses. It has to provide a better understanding of the studied topics. TOST is intended to give students an operating system inside view. It includes a multitasking operating system, a compiler for a simple PASCAL-similar programming language and a machine-language interpreter. TOST compiler prepares code-files of the user programs to be executed in concurrent mode. Main operating system parameters can be changed dynamically. The information in basic system tables, variables, memory allocation and addressing can be watched.*

Key words: *Simulators, Operating Systems, Parallel Programming*

INTRODUCTION

The Operating System (OS) is one of the most complicated software products have ever developed. All its basic algorithms are fundamental for the informatics at all. The OS solves problems which are area of interest of many other software applications with a particular purpose. Because of that the Operating Systems course, providing basic knowledge in computer systems functionality, is one of the fundamentals in computer specialist's education.

The common practice at the teaching process is the usage of a commercial OS for theoretical principles demonstration. It is not always suitable. The need of system safety and visual experimenting with basic system parameters necessitates applying of simulators. Unfortunately a bit of existing OS simulators are suitable enough. Some of the necessary requirements of an OS simulator used in teaching process are:

- to enable students to get an inside view of the basic modules and functionality of a standard OS;
- to support multitasking and to have possibility of implementing of mutual exclusion, synchronization and communication of concurrent processes;
- to allow a dynamic change (at any time) of basic OS parameters (Scheduling Strategy, Memory Management, Quantum Size, Page Size, etc.) [1], as well as those of the running processes (Priority, Remaining Time, etc.) [1]. This way the students have possibility to investigate how this modifications affect on the programs execution.
- to have an already known interface, to support a familiar programming language and to be user-friendly;
- to allow working on accessible hardware platforms.

The considering software system TOST is an integrated environment corresponding to a great extent to requirements mentioned above. It is used as a teaching tool in the "Operating Systems" and "Language Processors" courses at the Rousse University "Angel Kanchev". Any compatible with IBM PC (MS Windows) platform is suitable for system functionality.

The integrated environment is developed with a standard multi-window interface and includes the following main modules:

- a single-user multitasking OS;
- a text editor and compiler to a virtual processor code. There is a simple PASCAL-oriented programming language for user's programs definition. SEMAPHORE and SHERED data types and statements for parallel programming possibilities are added;
- a virtual processor emulator.

The system enables students:

- ✓ to create and run their own programs in concurrent mode;
- ✓ to watch system tables data records;
- ✓ to change dynamically the main OS parameters and those of the existing processes.

TOST INTERFACE AND MAIN FACILITIES

Access to the system functions is performed through an integrated multi-window interface with a main menu, shown on fig.1.

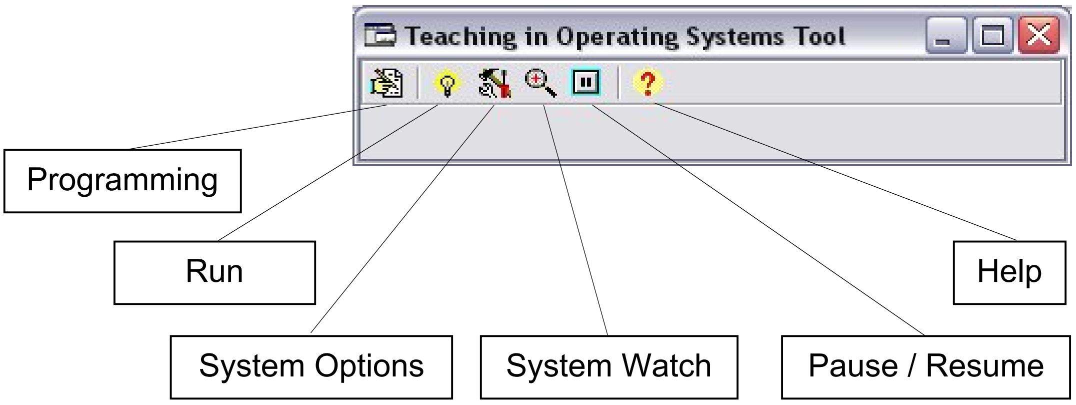
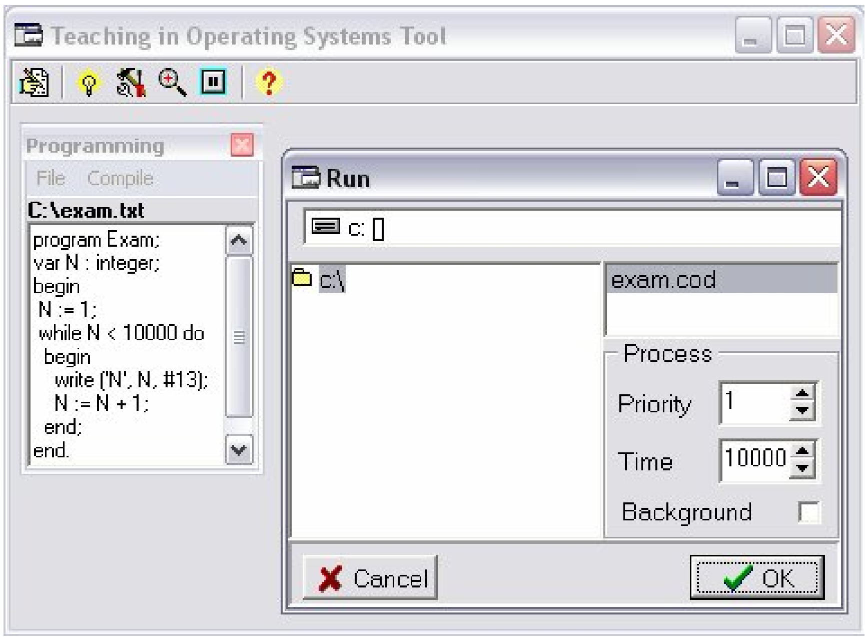


fig.1. TOST main menu

All main operations concerning user programs creating, editing, saving and compiling are concentrated in **Programming** item. After the successful compilation, the object program file is generated (with .COD extension) and it is ready to be running in the environment. A new process is started through **Run** at the main menu.

Figure 2 presents the general system view, when in an edit window is placed simple example program **exam.txt**, and the initial parameters of a creating process are set before running in another dialog box.



C:\exam.txt

```
program Exam;
var N : integer;
begin
 N := 1;
 while N < 10000 do
  begin
   write ('N', N, #13);
   N := N + 1;
  end;
end.
```

c:

c:\

exam.cod

Process

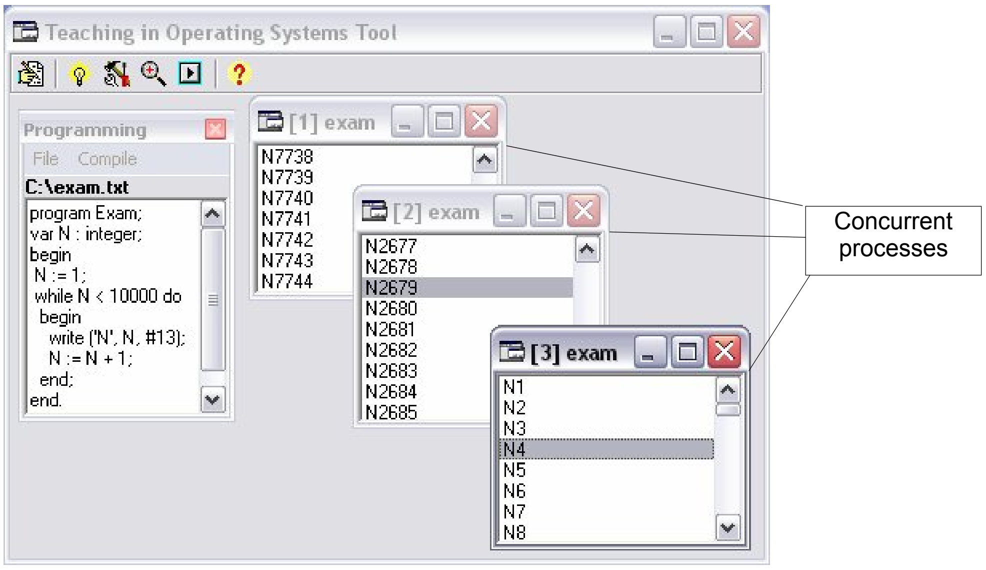
Priority 1

Time 10000

Background ☐

fig.2. Programming and Run

In the **Priority**, the initial process priority could be entered. The total time for process execution, entered in **Time,** is used in some scheduling strategies. Through **Background,** the user can specify the process visual mode. Repeated **Run** will cause starting of many concurrent processes (each in a separate window) as shown at fig.3.



```
program Exam;
var N : integer;
begin
 N := 1;
 while N < 10000 do
  begin
   write ('N', N, #13);
   N := N + 1;
  end;
end.
```

```
N7738
N7739
N7740
N7741
N7742
N7743
N7744
```

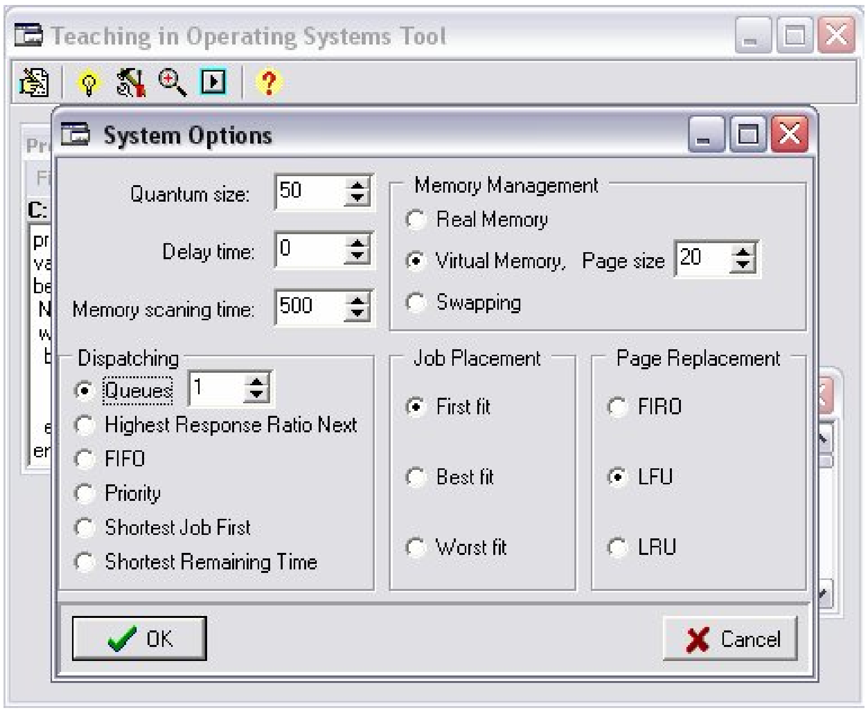
```
N2677
N2678
N2679
N2680
N2681
N2682
N2683
N2684
N2685
```

```
N1
N2
N3
N4
N5
N6
N7
N8
```

fig.3 Concurrent processes execution

The real multitasking allows editing, creating and running new programs while some other processes exist. When more detailed monitoring of running processes is required, through **Pause/Resume** a temporal "freezing" can be done.

The main OS parameters could be dynamically changed by **System Options** item of the main menu (fig.4).



Quantum size: 50

Delay time: 0

Memory scaning time: 500

Memory Management: Real Memory [✓ Virtual Memory, Page size 20] Swapping

Dispatching: [✓ Queues 1] Highest Response Ratio Next FIFO Priority Shortest Job First Shortest Remaining Time

Job Placement: [✓ First fit] Best fit Worst fit

Page Replacement: FIFO [✓ LFU] LRU

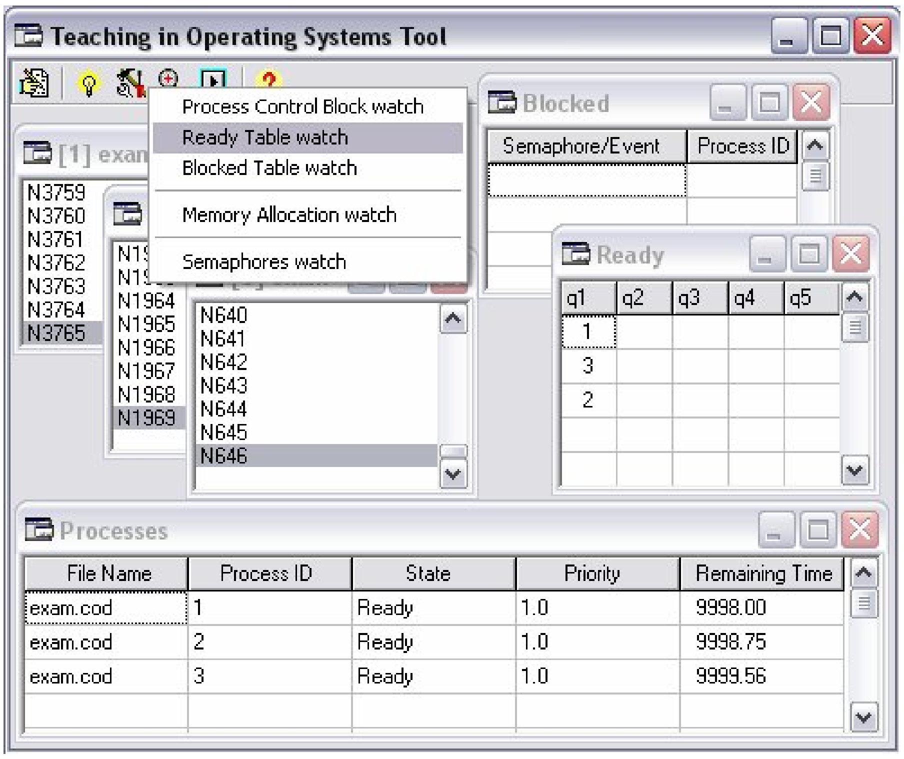
fig.4. System Options modification

In **System Options** the user at any time can change:

- system quantum - **Quantum Size [ms]**;
- delay pause in processes running for more detailed tracing - **Watch Delay time**;
- scheduling strategy – **Dispatching;**
- number of queues with ready processes - **Queues**;
- memory management strategy – **Memory Management**;
- page size of virtual memory - **Page size**;
- jobs placement strategy at real memory mode – **Job Placement**;
- page replacement strategy at virtual memory mode - **Page Replacement**;

These modifications of system parameters allow tracing in real time how each of them influences on started processes execution.

Watching the dynamics in the contents of some main system tables is another purpose of the integrated environment. These possibilities could be reached by **System Watch** item from the main menu (fig.5).



N3759
N3760
N3761
N3762
N3763
N3764
N3765

N1964
N1965
N1966
N1967
N1968
N1969

N640
N641
N642
N643
N644
N645
N646

Semaphore/Event	Process ID

q1	q2	q3	q4	q5
1				
3				
2				

File Name	Process ID	State	Priority	Remaining Time
exam.cod	1	Ready	1.0	9998.00
exam.cod	2	Ready	1.0	9998.75
exam.cod	3	Ready	1.0	9999.56

fig.5. Main System Tables Watch

The **Processes** table includes the current **Process Control Block** of each of the existing concurrent processes. The information is shown in five fields:

- **File Name** – object file name – file from which the process is started;
- **Process ID** – unique process identifier – an integer value;
- **State** – current process state – Ready, Running or Blocked;
- **Priority** – current process priority – a real value;
- **Remaining Time** – time, remaining to the process competition;

The **Ready** table presents information about the processes which are in state "Ready". According to the scheduling strategy it is possible to have up to 5 queues with ready processes with different behaviors (with more I/O or more computing operations).

The identifiers of blocked processes and the names of semaphores/events, relative to their blocking are recorded in the **Blocked** table**.**

Additional information about the memory allocation of processes is given by **Memory Allocation Watch**. When this submenu item is selected, the system periodically scans memory occupation and shows results in another window. The virtual memory pages addressing is displayed too and this allows real time observation of "temporally locality" and "space locality" events [1].

Semaphores Watch submenu item could be selected for semaphores values dynamic watching.

CONCLUSIONS AND FUTURE WORK

The presented system is quite opened and might be evolved on the following directions:

- developing its own file system with set of organizations and management strategies;
- adding a spooling-system and functions for peripheral devices control;
- adding complex data types and subroutines in the programming language;
- adding Pascal or C syntax optional usage.

Despite of some limitations, the integrated environment can be used for basic teaching in "Parallel Programming" too. Many classic example tasks for synchronization and mutually exclusion, like:

- ✓ "Dining philosophers";
- ✓ "Sleeping barber";
- ✓ and any variants of "Producer/Consumer" problem

could easy be demonstrated within TOST.

REFERENCES

[1] Tanenbaum, A, S., Modern Operating Systems, Eaglewood Cliffs, NJ, Prentice Hall, 2001.

ABOUT THE AUTHORS

Principal assistant. Tzanko Golemanov, Department of Computer Systems and Technologies, University of Rousse, Phone: +359 82 888 681, E-mail: TGolemanov@ecs.ru.acad.bg.

Principal assistant. Emilia Golemanova, Department of Computer Systems and Technologies, University of Rousse, Phone: +359 82 888 681, E-mail: EGolemanova@ecs.ru.acad.bg.