The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program
of the Lobachevsky State University of Nizhni Novgorod
among the world's research and education centers

Strategic initiative

Achieving leading positions in the field of supercomputer technology and high-performance computing

Computational mathematics and cybernetics faculty

APPROVED

Dean of the Computational mathematics
and cybernetics faculty

________________ V. Gergel

"_____"__________________2014.

Course program

OPERATING SYSTEMS, SUPPORT FOR PARALLELISM

Bachelor program "Fundamental Informatics and Information Technologies"

Specialization

General part

Professional block

Б3.Б.5

Degree

Bachelor

Education form

Full-time attendance

Nizhni Novgorod

2014

1. Course objective

The course's object is to master the basic concepts, methods and algorithms of operating systems architecture and functions concerned with parallel programs development. The subject includes models and algorithms used in implementations of various subsystems, application runtime environments and architecture of modern operating systems. Special attention is given to operation of multitask systems, multiprocess and multithreaded applications, which is critical for development of programs for multiprocessor / multicore systems.

Lecture sections propose studying of main concepts of organization of multitasking environment in operating systems (process and thread), CPU time allocation between threads idea and base algorithms, necessity of synchronization, low-level synchronization algorithms and high-level synchronization mechanisms usually provided by Oses.

Laboratory practice contains several tasks which illustrates considered operations with processes and threads in Windows and UNIX families of operating systems and also usage of standard synchronization mechanisms in them by the examples of standard synchronization problems.

The main objective of the course is to study the base operating systems concepts concerned with parallelism and supported at OS level.

The following tasks are stated:

1. Study of basic concepts concerned with parallel execution - process and threads.

2. Study of scheduling - CPU time allocation between threads concept and algorithms.

3. Study of synchronization idea, low-level algorithms and high-level mechanisms.

4. Mastering popular implementations of process/thread operations and synchronization mechanisms implementations.

2. Course position in the bachelor program

The course is developed for the 2-th year students and is given in the 4-th semester. This course belongs to the compulsory disciplines of the professional block.

The course is oriented on the students with some basic knowledge in the structured and modular programming and basic skills of C program development. Some knowledge in computer architecture is desirable.

3. Learning outcomes and requirements

In the framework of this course, the following **competencies** are formed:

- Possessing the general culture of thinking, the ability to perceive, compile and analyze information (General Competency 1 - GC1). Students will be able to:
    - construct oral and written arguments in a logical and clear manner.
- The ability of intellectual, cultural, moral, physical, and professional self-development and self-improvement (GC 2). Students will be able to:
    - constantly improve their professional and cultural level.
- The ability to understand and apply in practice the theory of information as a fundamental scientific basis of information technology (Professional Competency 1 – PC 1). Students will be able to:
    - understand the content side of the information process, know the techniques for sending, receiving, processing, analyzing and storing data.
- The ability to understand, develop and apply modern information technology (PC 4). Students will be able to:
    - understand the concepts and implement the functionality of the following core technologies:
        - at the level of in-depth knowledge:
            - development and principles of organization of operating systems;
            - parallel and distributed computing.
    - develop and use professionally the software for supporting information systems and processes, to be able to use modern instrumental computing equipment.
- The ability to conduct research (PC 5). Students will be able to:
    - develop new algorithmic and methodological and technological solutions;
    - collect, process and interpret the data of modern research necessary to develop approaches, decisions and conclusions on appropriate scientific and professional issues.
- The ability to conduct analytical activities (PC 8). Students will be able to:
    - analyze and select modern technologies and methodologies for implementing an information system.

As the result of education graduates from the course:

Graduates from the course will have knowledge of principles and mechanism of parallelism support at operating systems level..

Will know base concept, models and algorithms used in OS subsystems concerned with parallel programming.

Will Be Able To develop multiprocess and multithreaded applications for various operational environments.

Can use system software (compiler) for development of parallel programs.

4. Course outline

The course consists of 1 credit, 36 hours, including 8 lecture hours and 8 practice hours. Practice classes can be held as lab works (students carry out assignments step-by-step under supervision) or master class (supervisor demonstrates and explains step-by-step solutions). 20 hours are allocated for individual work. The authors encourage additional work.

4.1. Course outline

Course outline is as follows:

#	Module	Semester	Week	Module type				Assessment
				Lecture	Seminar	Lab	Individual work	
1	**Processes and Threads.**	4	1-2	2	–	–	2	Test
2	**Scheduling.**		3-4	2	–	–	2	Test
3	**Synchronization-1.**		5-6	2	–	–	2	Test
4	**Synchronization-2.**		7-8	2	–	–	2	Test
5	**UNIX process creation**		9-10	–	–	2	3	Test
6	**Windows process and thread creation**		11-12	–	–	2	3	Test
7	**Synchronization - 1. UNIX mechanisms.**		13-14	–	–	2	3	Test

8	**Synchronization - 2. Standard problems.**		15-16	–	–	2	3	Test
	TOTAL:			**8**	–	**8**	**20**	**Final assessment form – exam**

4.2. Course description

Course content is as follows:

1. Processes and Threads.

Definitions, attributes, thread execution state diagram, process and thread creation and termination.

2. Scheduling.

Criteria of scheduling algorithms comparison. Classification of short-term scheduling algorithms. Examples of algorithms: FIFO, SJN, SRT, RR, priority algorithms, Multi-Level Feedback Queue scheduling, Windows Scheduling.

3. Synchronization-1.

Critical resources and critical sections. Use of locking variables. Dekker's algorithm. Bakery algorithm. Test-and-set and swap instructions and active waiting.

4. Synchronization-2.

Dijkstra's semaphore primitives. Mutex. Monitor. Producer-Consumer and Readers-Writers problems.

5. UNIX process creation

UNIX system calls fork(), exec*(), exit(), wait(). Example #1 - New process creation, UNIX-style. Example #2 Creating a pipeline.

6. Windows process and thread creation

Process creation and termination in Windows. Example #3 – Process creation, Windows-style. Example #4 – Thread creation, Windows.

7. Synchronization - 1. UNIX mechanisms.

Example #5 – Multithreaded application, UNIX. Some UNIX synchronization mechanisms (POSIX semahpores, mutexes, conditional variables). Example #6 – Multithreaded application with mutual exclusion problem solved, UNIX, mutex. Example #7 – Multithreaded application with mutual exclusion problem solved, UNIX, semaphore.

8. Synchronization - 2. Standard problems.

Windows synchronization mechanisms (waitable objects, semahpores, mutexes). Example #8 – Readers-Writers problem, Windows. Example #9 – Producers-Consumers problem, Windows.

5. Learning technologies

During course we use the following learning technologies: lectures, lab works, individual work, assessment techniques. Powerpoint presentations for all lectures and practical lessons are used.

6. Individual work and assessment techniques

Individual work consists of mastering theoretical and practical material according to the given references, solving practical problems, and answering on the given questions. Individual work can be done in both classes and at home.

Control of individual work is performed by electronic tests. In the end of the program there is a final test.

6.1 Assessment forms

Monitoring of progress in studies is performed by tests in class that consist of assignments from the list of questions and practical problems (given below).

The final attestation is done based on the results of the final test. This test includes questions from all sections of the course.

6.2 Individual work: Questions and Practical problems

Module 1

1. Reasonably explain what method of starting new programs you prefer: UNIX-like or Windows-like
2. Complete (expand) process/thread creation and termination steps listed in the lecture.
3. Build a thread states diagram in a single-tasking system based on the thread states diagram in multitasking system.

Module 2

4. Suggest the algorithm for dynamical calculation of priorities which includes at least 3 statistical parameters of thread execution. Analyze and prove if “starvation” is possible by applying your algorithm.
5. Develop Windows application with thread affinity mask usage.

6. Build execution diagrams of 3 threads in a single-task system and build their execution diagram in a multitasking system with different scheduling algorithms. Determine the values for quality criteria of scheduling algorithms for the obtained diagram: the number of context switches (overhead), throughput, turnaround time, efficiency, wait time.

Module 3

7. For each requirement to the problem of mutual exclusion, build incorrect solution violating this requirement.
8. Modify the sequence of instruction execution in Peterson’s algorithm. What modifications keep the solution correct?
9. Explain the approach underlying Decker’s/Peterson’s algorithms and bakery algorithm.
10. Search for other CPU instructions (excluding Test&Set и Swap), which can be used to solve the problem of mutual exclusion.
11. Write a program implementing and using Peterson’s algorithm.
12. Write a program implementing and using the bakery algorithm.

Module 4

13. List the typical features of mutexes.
14. Implement semaphores using mutexes.
15. Implement semaphores using monitors.
16. Implement monitors using mutexes and/or semaphores.
17. Solve the « Readers-Writers » problem using monitors.
18. Write the solution of the «Readers-Writers» problem which will provide writer priority.

Module 5

19. Write a program which, executing a loop, requests a string from the user, creates a new process and execute a program inside it with command line from input string. If the first word of input string is “exit”, finish the program.
20. Write a program which executes a pipeline with 3 or more commands, for example:
    ```
    cat /etc/passwd | grep user | sort
    ls -la /dev | grep sd | sort | less
    ```
21. Write a program which creates the process tree and outputs PID for each of them (to deliver the data into the root process the single unnamed pipe shall be used).

Module 6

22. Write a program which, executing a loop, requests a string from the user, creates a new process and execute a program inside it with command line from input string. If the first word of input string is “exit”, finish the program.
23. Write a program which implements following operations:

   a. create new process with “Notepad.exe” program running inside it and use an additional command line parameter "C:\Windows\system.ini“ (or chose another existing file name);
   b. output the child process identifier;
   c. wait for 10 seconds;
   d. if notepad will not finish in this 10 seconds, terminate it;
   e. output the return code of Notepad.

24. Write a program and explain result of it’s execution. Program must implement following operations:
    a. define a global volatile integer variable GlobalVar with initial value 0;
    b. output the value of GlobalVar;
    c. create 10 threads;
    d. every thread must get address of GlobalVar as a function parameter;
    e. every thread must increment GlobalVar 100 million times;
    f. wait for all threads completion;
    g. output the final value of GlobalVar.

Module 7

25. Write the correct solution for the problem given in practice. Use your own synchronization mechanism based on one of the algorithms described in Lecture 3.
26. Write the correct solution for the problem described in this laboratory research using one of the mechanisms implemented in Windows.
27. Compare different program implementations from the performance point of view.

Module 8

28. Write a program which solves a “Readers-Writers” problem for UNIX operating system.
29. Write a program which solves a “Producers-Consumers” problem for UNIX operating system.
30. Study the “Dining philosophers” problem and write a program which solves it for Windows or UNIX.

6.4 Assessment criteria

Perfect	Correct answers on >95% of the number of test questions
Excellent	Correct answers on 80-95% of the number of test questions
Very good	Correct answers on 70-79% of the number of test questions
Good	Correct answers on 60-69% of the number of test questions

Satisfactory	Correct answers on 50-59% of the number of test questions
Unsatisfactory	Correct answers on 25-49% of the number of test questions
Bad	Correct answers on <25% of the number of test questions

7. References

7.1 Main literature

1. Abraham Silberschatz, Peter B. Galvin and Greg Gagne. Operating System Concepts. 8-th Edition.
2. Andrew S. Tanenbaum. Modern Operating Systems (3rd Edition).
3. Mark Russinovich, David Solomon and Alex Ionescu. Windows Internals (5th Edition) (Developer Reference).
4. Love R. Linux Kernel Development. SAMS. 2003.

7.2 Further reading

5. Mark Russinovich, David Solomon and Alex Ionescu. Windows Internals (5th Edition) (Developer Reference).
6. Jeffrey M. Richter, Christophe Nasarre. Windows via C/C++, 5th edition.
7. Robert Love. Linux Kernel Development (3rd Edition).
8. Daniel P. Bovet. Understanding the Linux Kernel, Third Edition.
9. Robert Love. Linux System Programming: Talking Directly to the Kernel and C Library.

7.3 Software and internet resources

10. www.vmware.com (VMWare Player)
11. www.msdn.com
12. www.sysinternals.com
13. www.tldp.org
14. www.linux.org

8 Course support

The following software and hardware are used during course study:

Hardware

Desktop computers.

Software

Windows 7, Microsoft Visual Studio 2005 or newer, VMWare Player 3.x, Linux (nearly any distribution), midnight commander, gcc compiler.

9 Authors

A. Linev.

Lab chief ____ Alexey Linev

Course program is discussed by Software Department members.

«_____» ________________ 2014; Document # _______

The head of the Programming Engineering Department, prof. ______________ Victor Gergel

Course program is approved by methodical commission of Computational Mathematics and Cybernetics Faculty of UNN,

«______» ________________ 2014; Document # _______

The head of the commission _______________________________ Natalia Shestakova