**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ СЕМЕНА КУЗНЕЦЯ**

**ЗАТВЕРДЖЕНО**
на засіданні кафедри
інформаційних систем.
Протокол № 1 від 22.08.2023 р.

**ПОГОДЖЕНО**
Проректор з навчально-методичної роботи

[illegible] ________________ Каріна НЕМАШКАЛО

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ * ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ №02071211

# СИСТЕМНЕ ПРОГРАМУВАННЯ

**робоча програма навчальної дисципліни (РПНД)**

| | |
|---|---|
| Галузь знань | **12 "Інформаційні технології"** |
| Спеціальність | **121 "Інженерія програмного забезпечення"** |
| Освітній рівень | **перший (бакалаврський)** |
| Освітня програма | **"Інженерія програмного забезпечення"** |

| | |
|---|---|
| Статус дисципліни | **вибіркова** |
| Мова викладання, навчання та оцінювання | **англійська** |

Розробник:
к.т.н., доцент підписано КЕП Дмитро ГОЛУБНИЧИЙ

Завідувач кафедри
інформаційних систем [illegible] ________________ Дмитро БОНДАРЕНКО

Гарант програми [illegible] ________________ Олег ФРОЛОВ

**Харків**
**2024**

[illegible]

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
SIMON KUZNETS KHARKIV NATIONAL UNIVERSITY OF ECONOMICS

**APPROVED**
at the meeting of the department
information systems

Protocol № 1 of 22.08.2023

**AGREED**
Vice-rector for educational and methodical work

[illegible] ________________ Karina NEMASHKALO

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ СЕМЕНА КУЗНЕЦЯ
№02071211

# SYSTEM PROGRAMMING
**Program of the course**

| | |
|---|---|
| Field of knowledge | **12 "Information Technology"** |
| Specialty | **121 "Software engineering"** |
| Study cycle | **first (bachelor)** |
| Study programme | **"Software engineering"** |

| | |
|---|---|
| Course status | **elective** |
| Language | **English** |

| | | |
|---|---|---|
| Developers:<br>PhD (Technical sciences),<br>Associate Professor | digital signature | Dmytro HOLUBNYCHYI |
| Head of Information systems department:<br>Ph.D. (Technical sciences),<br>associate professor | [illegible] | Dmytro BONDARENKO |
| Head of Study Programme<br>Ph.D. (Technical sciences),<br>associate professor | [illegible] | Oleg FROLOV |

**Kharkiv**
**2024**

[illegible]

# INTRODUCTION

Today's economic conditions demand comprehensive use of the latest information technologies from economic management specialists. The broad capabilities of computerized means in matters of collecting, processing and issuing the necessary information can significantly improve the quality of economic calculations, make the process of justifying economic decisions more effective. But the successful use of a powerful computerized tool is impossible without a clear idea of the peculiarities of the functioning of all its constituent parts, and this, in turn, requires solid knowledge of the processes that take place in the operating system at the level of resource management during their work.

Knowledge of the basics of building operating systems is becoming more and more relevant, since the trends in the development of computer technology indicate that, on the one hand, the complexity and functionality of computer technology are constantly and rapidly growing, and on the other hand, there is a constant trend towards the personification of this complex techniques That is, the task of maintaining a personal computer in working condition, adjusting the operation of its software and configuration, timely upgrade (patch, crack) increasingly becomes a problem not for professionals, but for the specific user of this personal computer.

The need to study system programming is determined by the emergence of new mechanisms of interaction between system and user software, which are required for compiling programs from common programming languages.

The purpose of the course "System programming" is to provide students with higher education with a system of special knowledge to master the theoretical foundations of construction, principles of design, configuration and application of various modern operating systems that ensure the organization of computing processes in corporate information systems of economic, management, production, scientific and other purpose, as well as providing practical skills for automating day-to-day administration tasks.

The objectives of the course are:

- mastering the principles of construction, purpose, structure, function and order of creating system programs for various operating systems, their subsystems, resource management mechanisms;
- mastering the basic methods of diagnostics, recovery, monitoring and optimization of operating system components through the use of system utilities, specialized libraries, etc.;
- mastering the skills of interaction with objects of the operating system by studying their characteristics and methods of operation through the use of system utilities.

The subject of the course is modern theoretical concepts and methodologies, principles of functioning, interaction of system components as part of the operating system.

The object of the course is the process of creating special software for computer systems, taking into account the peculiarities of the functioning of operating systems.

The learning outcomes and competence formed by the course are defined in the table. 1.

Table 1

**Learning outcomes and competencies formed by the course**

| **Learning outcomes** | **Competencies** |
|---|---|
| LO 14 | GK 2 |
| LO 01 | GK 6 |
| LO 04 | SK 04 |
| LO 07 | SK 06 |
| LO 15 | SK 10 |

where, LO 01. Analyze, purposefully search for and choose information and reference resources and knowledge necessary for solving professional tasks, taking into account modern achievements of science and technology.
LO 04. Know and apply professional standards and other regulatory documents in the field of software engineering.
LO 07. To know and apply in practice the fundamental concepts, paradigms and basic principles of functioning of linguistic, instrumental and computing tools of software engineering.
LO 14. Apply in practice instrumental software tools for domain analysis, design, testing, visualization, measurement and documentation of software.
LO 15. It is motivated to choose programming languages and development technologies to solve the tasks of creating and maintaining software.
GK 2. Ability to apply knowledge in practical situations.
GK 6. Ability to search, process and analyze information from various sources.
SK 04. Ability to formulate and ensure software quality requirements in accordance with customer requirements, specifications and standards.
SK 6. Ability to analyze, select and apply methods and tools to ensure information security (including cyber security).
SK 10. The ability to accumulate, process and systematize professional knowledge about creating and maintaining software and recognizing the importance of lifelong learning.

## COURSE CONTENT

### Content module 1. System programming in Windows

#### Topic 1. Windows application architecture

**1.1.**Framework Windows application. Window class. The main function. Creating a window. Creating windows using existing classes. Definition of the WNDCLASS structure. Window class registration. Message handling in a window function. Windows support functions. Attaching window class data to the window.

Changing the appearance of the window. Application programming interface.

**1.2.** The main function of the application. Data types used in Windows. Message processing cycle. Sources of messages. Message interception functions. Message queues.

**1.3.** Management bodies. Text editors. Scroll bars.

**1.4.** User system interfaces. Interface of graphic devices. Introduction of objects to the context of devices. Get device context descriptor. Display context. Class context. General context. Private context. Display context functions. Graphical user interface. Multi-window interface.

**Topic 2. Message interception mechanisms**

**2.1.** Basic message interception mechanisms. Windows-hooks. Loading hooks. Tasks and organization of message interception procedures. Interception functions. Hook installation and uninstallation.

**2.2.** Specialized hooks. Local hooks. Remote hooks. System hooks. Types of hooks. Messages for working with hooks. Defining arguments for hook handlers.

**Topic 3. Processes, threads and means of interprocess interaction in Windows**

**3.1.**Processes. Process functions. Application IDs. Process command line. Changing environments. Process status. Error handling. Working process directories. Creating and ending processes. Protection of processes from unprofitable code. Error and exception handling.

**3.2.** Streams Conditions for creating streams. Stream stack. Flow state. Flow execution periods. Creation and termination of threads. Allocation of processor time between threads. Changing the priority class of a thread. Delaying and resuming thread execution.

**3.3.** Flow planning and dispatching. Types of planning. Planning strategies. Displacing and non-displacing multitasking. Flow scheduling algorithms. Quantization. Flow planning in real-time systems.

**3.4.** Basic mechanisms of interprocess interaction. Interprocess interaction based on shared memory. Message transmission technologies.

**3.5.**Basic mechanisms of thread synchronization. Semaphores. Mutexes. Block read-write. Synchronization according to the barrier principle. Thread interaction in Windows. Software interaction interface. Distributed memory methods. Methods of message transmission. Technology of displayed memory.

**Topic 4. Fundamentals of Windows operating system security**

**4.1.** The main tasks of ensuring security. Basic concepts of cryptography. Concept of cryptographic algorithm and protocol. Cryptosystems with a secret key. Public key cryptosystems. Hybrid cryptosystems. Digital signatures. Certificates.

**4.2.** Principles of authentication and access control. Types of objects that are protected. Formation of access control lists. Implementation of protection of personal objects. User accounts. Audit General principles of audit organization. Windows Event Log.

**4.3.** Principles of data encryption on file systems. Creating a crypto provider. Windows Encrypting File System. Network data security. Information protection at the network level.

**Content module 2. System programming in Linux**

**Topic 5. Linux operating system architecture**

**5.1.**Architecture of distributions of Linux operating systems. Multilevel systems. Microkernel architecture. Basic kernel mechanisms. Resource managers. System call interface. Operating system resources. Hardware dependency and portability of the operating system. Graphical interfaces.

**5.2.** The kernel of the operating system and its functions. Auxiliary modules of the operating system. Kernel in privileged mode and user mode. Exchange between applications when using the kernel in privileged mode.

**5.3.** File system management. Implementation of Linux file systems. Standard file system directories. Device files. Mounting. File managers in Linux.

**5.4.**Commands for managing files and folders. Commands for working with files. Commands for working with folders. The ln command and its options. Search for files in the console. Help viewer man. Archiving. Tar and gzip utilities.

**5.5.**Linux command line. An overview of Linux shell commands. The bash shell. Features of work. Access the command line. Command line symbols. Checking conditions. Operations. Cycles and branching. Internal and external teams. System administration teams. Substitution of commands.

**5.6.**Basics of working with scripts. Basics of scripting. Using structured commands. Processing user input. Data presentation. Script management. Creating functions. Writing scripts for graphical desktops. General information about the sed and gawk editors. Working with other command interpreters.

**Topic 6. Creation, compilation and arrangement of programs in Linux**

**6.1.**Creating programs in Linux. Source code. Compilation. Layout. Multi-file projects.

**6.2.**Self-assembly. An overview of Linux autocompilers. The make utility. Basic MakeFile syntax. Make constants. Recursive call to make. Obtaining additional information.

**6.3.** Environment. The concept of environment. Reading and modifying the environment. Cleaning the environment.

**6.4.** The concept of I/O in Linux. Library input-output mechanisms of the C language. The concept of low-level input-output. Console I/O. I/O in C++.

**6.5.**Basic input-output operations. Creating a file: create(). Opening a file: open(). Closing the file: close(). Reading a file: read(). Writing a file: write(). Random access: lseek().

**Topic 7. Basics of multitasking in Linux**

**7.1.** The basics of multitasking in Linux. Library approach: system(). Processes in Linux. Process tree. Obtaining information about the process.

**7.2.** Basic multitasking. Concept of fork: fork(). Transfer of control: execve(). The exec() family. Waiting process: wait().

**7.3.** The concept of threads in Linux. Create a thread: pthread_create(). Thread Termination: pthread_exit(). Waiting for thread: pthread_join(). Getting information about a thread: pthread_self(), pthread_equal(). Canceling a thread: pthread_cancel(). Obtaining additional information.

**7.4.** Advanced multitasking. Process compliance: nice(). The wait() family. Zombie.

**Topic 8. Methods of interprocess interaction in Linux**

**8.1.** An overview of methods of interaction between processes in Linux. General information about interactions between processes in Linux. Local methods of interaction between processes. Remote interprocess communication.

**8.2.** Signals. The concept of a signal in Linux. Sending a signal: kill(). Signal handling: sigaction(). Signals and multitasking.

**8.3.** Using shared memory. Memory allocation: shmget(). Activating sharing: shmat(). Disable sharing: shmdt(). Memory usage control: shmctl(). Use of semaphores. Traffic light control.

**8.4.** Using shared files. Placing a file in memory: mmap(). Freeing memory: munmap(). Synchronization: msync().

**8.5.** Channels. Create a pipe: pipe(). I/O redirection: dup2(). Create a named channel. Read, write and close FIFO.

The list of laboratory studies in the course is given in table 2.

Table 2

**The list of laboratory studies**

| Name of the topic and/or task | Content |
|---|---|
| Topic 1. | Study of the structure of the Windows application |
| Topic 2. | Study of message interception mechanisms in multi-window documents |
| Topic 3. | Study of processes and flows |
| Topic 3. | Study of means of data exchange between processes |
| Topic 4. | Investigating operating system security tools using CryptoAPI |
| Topic 5. | Exploring the capabilities of shell programming |
| Topic 6. | Study of ways to create programs in Linux |
| Topic 7. | Exploring ways to multitask programming in Linux |
| Topic 7. | Process management in Linux |
| Topic 8. | Study of mechanisms of interaction between processes in Linux |

The list of self-studies in the course is given in table 3.

Table 3

**List of self-studies**

| Name of the topic and/or task | Content |
|---|---|
| Topic 1 – 8 | Studying lecture material |
| Topic 1 – 8 | Preparation for laboratory classes |
| Topic 1 – 8 | Preparation for the exam |

The number of hours of lectures, laboratory classes and hours of self-study are given in the technological card for the course.

## TEACHING METHODS

In the process of teaching the course, in order to acquire certain learning outcomes, to activate the educational process, it is envisaged to use such teaching methods as:

Verbal (lecture-discussion (Topic 1, 3, 5, 7, 8), problematic lecture (Topic 2), lecture-visualization (Topic 4, 6)).

Visual (demonstration (Topic 1 – 8)).

Laboratory work (Topic 1–8), case studies (Topic 1 – 3, 7).

## FORMS AND METHODS OF ASSESSMENT

The University uses a 100-point cumulative system for assessing the learning outcomes of students.

**Current control** is carried out during lectures, laboratory classes and is aimed at checking the level of readiness of the student to perform a specific job and is evaluated by the amount of points scored:

– for courses with a form of semester control as an exam: maximum amount is 60 points; minimum amount required is 35 points.

**The final control** includes current control and an exam.

**Semester control** is carried out in the form of a semester exam.

***The final grade in the course*** is determined:

– for disciplines with a form of exam, the final grade is the amount of all points received during the current control and the exam grade.

During the teaching of the course, the following control measures are used:

Current control:

defense of laboratory work (40 points);

written control work (testing) (20 points).

Semester control: Grading including Exam (40 points)

An example of an exam card and assessment criteria.

### An example of an examination card

Simon Kuznets Kharkiv National University of Economics
First (bachelor) level of higher education
Specialty "Software Engineering"
Educational and professional program "Software engineering"
Semester V
Course "System programming"

**EXAM CARD**

**Task 1** (heuristic, 20 points).

Develop a system Windows application and provide a listing of the software code of all necessary modules that must perform the following actions:
Create an application in which to set the filter function on the WH_CBT hook with code of type HCBT_ACTIVATE. Demonstrate and log all actions of the installed hook. Accompany the operation of the program by providing an output of messages about the installed hook.

**Task 2** (heuristic, 20 points).
Develop a system Linux application and provide a listing of the software code of all necessary modules that must perform the following actions:
Write an application that determines the classes, senior, junior numbers of the main peripheral devices implemented in the system: keyboard, mouse, USB-drive, printer, video adapter, CD/DVD drive (if available), cardreader, etc.

Protocol No. ____ dated "___"__________20___ was approved at the meeting of the Department of Information Systems

Examiner PhD, Associate Professor Holubnychyi D.

Chief department PhD, Associate Professor Bondarenko D.

**Assessment criteria**

The final marks for the exam consist of the sum of the marks for the completion of all tasks, rounded to a whole number according to the rules of mathematics.

The algorithm for solving each task includes separate stages that differ in complexity, time-consumingness, and importance for solving the task. Therefore, individual tasks and stages of their solution are evaluated separately from each other as follows:

**Task 1 (heuristic).**

The peculiarity of the first task is its execution under the control of the Windows operating system. The task is devoted to the development of system software for the given task and to provide a listing of the software code of all necessary modules. A program is compiled in the C++ programming language. The acquirer must create a project in the Visual Studio environment. The main goal of solving this problem is to check the practical skills of the acquirer to use the application programming interface to exchange data between the program and the operating system. At the same time, the acquirer is allowed to use the existing reference literature. After checking the program, the applicant receives K1 points according to the following requirements (Table 4).

Table 4

**Generalized evaluation criteria for heuristic tasks**

| Points K1 or K2 | Requirements |
|---|---|
| 0 | The applicant does not have the educational material, there is no answer to the question |
| 1 – 2 | The acquirer does not have the educational material, the program is missing or does not start at all. |
| 3 – 6 | The applicant possesses educational material at the elementary level of knowledge and reproduction of individual facts and fragments. The program, supporting files, and algorithm scheme are satisfactorily executed, there are significant spelling or syntax errors, and there are no comments. The program partially meets the task. In the case of its implementation, the main window is created after minor modifications, but most of the functions for working with it are not implemented. During operation, the program either freezes or crashes. |

| Points K1 or K2 | Requirements |
|---|---|
| 7 – 10 | The acquirer at the level of memorization reproduces the main provisions of the educational material. The program, supporting files and algorithm scheme are well done, there are minor spelling or syntax errors, no comments. The presence of unused variables, private functions and procedures is allowed. The program partially or not fully meets the task. If it is executed, the main window, window procedure, information about the author are created, but the rest of the functions for working with them are not implemented. |
| 11 – 14 | The applicant reveals the essence of the main provisions of the educational material. The program, auxiliary files, and the scheme of the algorithm are well executed, without spelling and syntactic errors, have a minimum number of comments, and the design style is present. The presence of unused variables, private functions and procedures is allowed. The program partially or not fully meets the task. In case of its execution, the modules mostly function correctly, but it needs additional finishing. |
| 15 – 17 | The applicant reveals the essence of the main provisions of the educational material. The program, auxiliary files, and the scheme of the algorithm are well executed, without spelling and syntactic errors, have a minimum number of comments, and the design style is present. The presence of unused variables is allowed. The program meets the task, but the absence of minor auxiliary functions that make it easier to work with it is allowed. In the case of its execution, the main modules function correctly, but there are some insignificant inaccuracies in the execution of the task. |
| 18 – 20 | The applicant has generalized knowledge of the organization of operating systems and applies them in practical work. The program and supporting files are perfectly executed, without spelling and syntax errors, have a clear structure, a certain number of comments, are designed at the appropriate level and do not contain unnecessary variables, inactive functions, procedures. It fully corresponds to the task and the case of execution - all modules function correctly. |

**Task 2 (heuristic).**

The peculiarity of the second task is its execution under the control of the Linux operating system. The task is devoted to the development of system software in the Linux operating system. List the software code of all necessary modules. A program is compiled in the C++ programming language. The main goal of solving this problem is to check the practical skills of the acquirer to use the application programming interface to exchange data between the program and the operating system. At the same time, the acquirer is allowed to use the existing reference literature. After checking the program, the applicant receives K2 points according to the following requirements (Table 4).

## RECOMMENDED LITERATURE

### Main

1. Галісеєв Г.В. Системне програмування: навч. посіб. / Г.В. Галісеєв. – Київ: Університет "Україна", 2019. – 113 с.

2. Підручник з предмету: Системне програмування. - Дніпро: ДФК, 2024. - 255 с. – URL: https://library.kre.dp.ua/Books/2-4 kurs/Системне програмування/Системне програмування C++.pdf

3. Черевик В.М. Операційна система Linux: принципи роботи з файловою системою. Навчальний посібник / В.М. Черевик, Л.І. Танцюра, С.С. Коротков, В.О. Сосновий. – Київ: ДУТ, 2021. – 147 с.

4. Мосіюк О. О. Операційні системи та системне програмування: навчально-методичний посібник // О. О. Мосіюк, А. Л. Федорчук. – Житомир: Вид-во ЖДУ ім. Івана Франка, 2022. – 76 с.

5. Tanenbaum E. Modern operating systems / E. Tanenbaum, H. Boss. – New Jersey: Pearson Prentice-Hall, 2020. – 1120 p.

6. Silberschatz A. Operating System Concepts / A. Silberschatz, G. Gagne, P.B. Galvin. – New Jersey: Wiley, 2021. – 1040 p.

7. Love R. Linux. System programming: Talking Directly to the Kernel and C Library / R. Love. – Newton: O'Reilly, 2023. – 448 p.

8. Schotts W. The Linux Command Line: A Complete Introduction / W. Schotts. – San Francisco: No Starch Press, 2022. – 480 p.

9. Suehring S. CompTIA Linux+ Practice Tests: Exam XK0-005 / S. Suehring. – New Jersey: Sybex-Wiley, 2022. – 1352 p.

## Additional

10. Граннеман С. Linux. Кишеньковий довідник / С. Граннеман. – Київ: Діалектика, 2019. – 464 с.

11. Dave T. Shell Scripting. Linux, OS X and Unix / T. Dave, P. Brandon. – San Francisco: No Starch Press, 2019. – 392 p.

12. Cooper M. Advanced Bash Scripting Guide - Volume 1: An in-depth exploration of the art of shell scripting / M. Cooper. – Independently published, 2019. – 582 p.

13. Uzayr S-b. Linux: The Ultimate Guide / Sufyan bin Uzayr. – Boca Raton: CRC Press, 2022. – 305 p.

14. Stollings V. Operating system / V. Stollings. – Washington: Pearson, 2020. – 1264 p.

15. Hud O. The Project Management Information System in Linux / O. Hud, O.Veres //COLINS'2020, Volume II: Workshop. – Lviv: Ukraine, 2020. – Pp. 270 – 273.

## Information resources

16. The official website of the Linux Ubuntu OS developers [Electronic resource]. – Access mode: https://ubuntu.com/.

17. Linux [Electronic resource]. – Access mode: https://www.linux.org/.

18. Mizyuk O. Guide to Linux [Electronic resource]. – Access mode: https://linuxguide.rozh2sch.org.ua.

19. Windows [Electronic resource]. – Access mode: http://windows.microsoft.com/ru-ru/windows/home.

20. Holubnychyi D. " System programming " [Electronic resource]. – Access mode: https://pns.hneu.edu.ua/course/view.php?id=4909