| | | | | | |
|---|---|---|---|---|---|
| Course Title | Systems Programming | | | | |
| Course Code | ACSC372 | | | | |
| Course Type | Compulsory | | | | |
| Level | BSc (Level 1) | | | | |
| Year / Semester | 2nd / 3rd (Spring) | | | | |
| Teacher's Name | Chrysostomos Chrysostomou | | | | |
| ECTS | 6 | Lectures / week | 2 | Laboratories/week | 2 |
| Course Purpose | This course aims to familiarize students with the concepts and principles underlying the field of systems programming in the UNIX operating system environment, and to enable students develop the skills related to elementary and advanced UNIX shell commands, shell environment and programming, system calls for Iinput/Output, process environment and control, signals handling, inter-process communication and socket programming. The role of systems programming in the UNIX environment is emphasized through practical work carried out by developing system-level programs in C programming language, performing various important tasks using UNIX shell commands and formulating shell scripts using structured shell programming. | | | | |
| Learning Outcomes | By the end of the course, the students are expected to:<br>1. recall operating system concepts, and explain the basic elements of the Kernel;<br>2. outline the structure and organization of the UNIX operating system;<br>3. implement shell configuration and programming, by performing tasks using UNIX shell commands and formulating shell scripts using structured shell programming;<br>4. examine and illustrate process creation, communication and management by generating system calls, signals and interrupts;<br>5. recognize and implement low level File Input/Output;<br>6. design programs for inter-process communications;<br>7. develop client-server models with the aid of socket programming. | | | | |
| Prerequisites | ACSC183, ACSC271 | Co-requisites | None | | |
| Course Content | • **UNIX System Overview:** UNIX operating system overview. Access to the Resources of a Computer. Basic Elements of the Kernel. System calls. Shell overview / Shell programming. UNIX history.<br>• **UNIX commands:** Elementary and advanced UNIX shell commands. Directories structure, Files and directories manipulation, Files examination, Input/Output streams, Files redirection, Pipes, Access permissions, Regular expressions, Process control. | | | | |

| | |
|---|---|
| | • **Bash Shell Environment:** Introduction to bash shell. Shells evolution. Environment and variables. Shell initialization files. Shell variables. Default global variables. User shell variables. Creating, deleting and exporting variables. Built-in shell commands. Executing shell scripts. Read user input data. Command line arguments. Special characters and quotes. Printing in the shell.<br>• **Bash Shell Programming:** Exit status code. Conditional statement if. Relational operators. File testing. Command line parameters. Logical operators. Arithmetic expressions. The case statement. The while loop and the until loop. Commands break and continue. The for loop. Shell functions. Parameters in functions. Scope of function variables. Initialization and processing of arrays. Advanced Shell Programming: Debugging bash scripts, Catching signals, Pipes and devices, Implementation of pipes with files, Implementation of pipes with two variables, The File descriptor table, Standard in/out/err, Devices, Input/Output redirection.<br>• **File Input/Output:** Error handling in C. Introduction to files and file system. File types in UNIX. How files are stored (Partitions, i-Nodes, Blocks). Example of finding a file using the i-nodes. File processing operations. Standard I/O vs. System Calls (Low-level I/O). System calls advantages and disadvantages. System calls for I/O.<br>• **Process Environment & Control:** Process Identifiers. Memory layout of a process. Creation of processes (When it is needed, What is not inherited). Environment Variables. Orphan process vs. Zombie process. Processes and files. Waiting of processes. How to avoid zombie processes.<br>• **Processes and Signals:** exec() Functions. System call system(). Signals Handling. Signals and Processes. Asynchronous avoidance of Zombie processes using Signals. System call alarm(). Inter-process signals. System call kill().<br>• **Inter-Process Communication:** Processes and Files. System calls dup() & dup2(). Inter-Process Communication (IPC) – Introduction. IPC with Pipes. System call pipe(). IPC with Named Pipes (FIFO). FIFOs in the shell. FIFOs in C. Pipes vs. FIFO.<br>• **Socket Programming:** Socket definition. Ports. Port numbers assignment. Socket domain families. Socket types. Client/Server TCP connection. Basic server functions. Basic client functions. Byte-order transformations. Socket descriptors. Socket creation. Socket address formats, Associating addresses with sockets. Listening to incoming connection requests. Accepting incoming connection requests. Connection establishment. Data transfer. Closing the connection. IP address conversions. More miscellaneous functions. Client/Server Example.<br>• **Laboratory Work:** Develop system-level software in C programming language, perform various important tasks using UNIX commands and formulate shell scripts using structured shell programming. |
| Teaching Methodology | Students are taught the course through lectures by means of computer presentations. Lectures are supplemented with laboratory work and homework that consist of simple and complex tasks performed using UNIX shell commands, shell scripts formulated using structured shell |

| | |
|---|---|
| | programming, and system-level programs in C programming language developed, aiming to help students develop practical skills by illustrating the systems programming concepts taught at lectures.<br>Lecture/Laboratory notes and presentations are available for students to use in combination with the textbooks and references, through the university's e-learning platform. |
| Bibliography | Textbook:<br>• Richard W. Stevens, Stephen A. Rago, ***Advanced Programming in the UNIX Environment***, Addison-Wesley, 3rd Ed., 2013<br>References:<br>• Kay Robins, Steven Robins, ***UNIX Systems Programming: Communication, Concurrency, and Threads***, Prentice Hall, 2nd Ed., 2016<br>• William Stallings, ***Operating Systems: Internals and Design Principles***, Pearson, 9th Ed., 2018<br>• Notes on UNIX commands & system utilities, Bash shell programming, and Socket Programming – all available on the university's e-learning platform |
| Assessment | The assessment of the course includes one written test and a final written exam with programmatically (using UNIX shell commands, structured shell programming and C programs) problem-solving questions. Laboratory work consists of practical problems aiming to help students understand and illustrate the systems programming concepts taught at lectures, one lab test and homework requiring students to develop system-level programs in C programming language, perform various important tasks using UNIX shell commands and formulate shell scripts using structured shell programming.<br>The weights for each assessment component are:<br>• Lab Work and Homework: 15%<br>• Tests: 25%<br>• Final Exam: 60% |
| Language | English |