Course unit title:	Programming in Unix-like Environments
Course unit code:	CSC326
Type of course unit: **(Compulsory/optional)**	Compulsory (Foundation)
Level of course unit: **(First, second or third cycle)**	Bachelor (1st cycle)
Year of study:	3
Semester when the unit is delivered:	5
Number of ECTS credits allocated:	6
Name of lecturer(s):	TBA

Learning outcomes of the course unit:

Upon succesful completion of this course students should be able to:

- Discuss basic Linux concepts.
- Compose intermediate to advanced level C++ code.
- Assemble computer programs that utilize memory effectively.
- Use stand alone debuggers to find persistent errors in source code

Mode of delivery:	Face- to- face
Prerequisites and co-requisites:	CSC205
Recommended optional program components:	None

Course Contents:

Objective:

To introduce the concepts of Unix-like operating systems, demonstrate the concepts of the C++ programming language, further the understanding of memory manipulation of personal computers, and introduce tools and techniques of debugging.

Description:

Introduction:
Linux basic concepts – file and directory management, line and screen editors, changing file and directory attributes, differentiation between binary and text files, creating text files from the command line, familiarization with the system files of Linux, basic c-shell scripting.

Overview of C++:
Makefiles, constant and variable types, expressions and operators, flow control structures, function definitions, header files, separate compilation and linking, understanding the compiling process, understanding the linking process.

Object oriented programming with C++:
Class definitions, inheritance, encapsulation, polymorphism and operator overloading using C++ structures, comparison between object-based and object-oriented programming.

Memory Management
Pointers (referencing and dereferencing), memory models to accommodate pointers, pointer arithmetic, argument passing (by value and by reference), pointers to pointers, pointers as arrays, pointers to objects, abstract data structures, file processing, dynamic object creation and deletion.

Command line compilation and debugging:
Basic debugging using cout statements. Using debuggers: getting to know gdb, using gdb to identify errors, breakpoints and watches.

Recommended or required reading:	Steve Oualline, PRACTICAL C++ PROGRAMMING O' Reilly Press Matt Welsh, Matthias Kalle Dalheimer, RUNNING LINUX Terry Dawson, Lar Kaufman, O' Reilly Press Michael K. Johnson, Erik W. Troan, LINUX APPLICATION DEVELOPMENT, Addison-Wesley Professional
Planned learning activities and teaching methods:	Class Instruction 42 Hours Consultation/Computer Lab 15 Hours
Assessment methods and criteria:	Examinations 50% Assignments/ Class Participation 50% 100%
Language of instruction:	English
Work placement(s):	No
Place of Teaching:	Theoretical Part: Regular Classroom European University Cyprus, Nicosia Practical Part: Computer Laboratory European University Cyprus, Nicosia