Systems Architecture

Academic Year: (2020 / 2021) **Review date: 08-07-2020**

Department assigned to the subject: Telematic Engineering Department

Coordinating teacher: IBAÑEZ ESPIGA, MARIA BLANCA

Type: Compulsory ECTS Credits : 6.0

Year : 2 Semester : 1

Branch of knowledge: Engineering and Architecture

REQUIREMENTS (SUBJECTS THAT ARE ASSUMED TO BE KNOWN)

Programming and Systems Programming

OBJECTIVES

The student must be able to design a software system using the C programming language. The system must contain non-trivial data structures, dynamic memory management and use engineering techniques to translate a set of high-level constraints, derived from a hypothetical industrial stage, in a robust application.

The student must be able to freely use the following tools used in industrial environments: a compiler with the options to generate different versions for debugging and analyse the messages that are obtained while developing an application, an integrated development environment to implement the system software, a cross compiler to create versions of an application, and tools for analysing memory behaviour in an application.

The student must be able to: work effectively in a team in the execution of a project consisting of the design of a software application, collaboratively generate ideas in a team and optimize their performance to meet the project requirements, and effectively divide tasks among team members.

The student must be able to: learn autonomously, manipulate the different sources of information, generate concise information about the tasks achieved, manage personal work time, and effectively present the results derived from their work.

DESCRIPTION OF CONTENTS: PROGRAMME

The programme is divided into the following blocks:
1. The C programming language
1.1. Basic data types and flow constructions
1.2. Structure of a C application. The pre-processor, division in files and creating an executable.
1.3. Pointer manipulation.

2. Dynamic memory management in C
2.1. Dynamic data structures
2.2. Memory leaks
2.3. Concurrent tools
2.4. Tools for detecting memory leaks

3. Architecture of the Linux
3.1. Kernel, processes, and filesystem
3.2. Main libraries
3.3. Concurrency

4. Team project design
4.1. Conflicts and their resolution
4.2. Project development

LEARNING ACTIVITIES AND METHODOLOGY

The activities used to underpin the competences and the skills in the course are (preceeded by the reference to the program objectives):

- Exercises covering the following topics: design the most appropriate data structure for a functionality in an application, write code fragments to manipulate data structures, read/write fields, process data, etc, calculate the amount of memory occupied by different data structures (PO: a).

- During the lab sessions code fragments are written, compiled, linked and executed using different compiler options and detect, analyze and correct these programs using the debugger (PO: b).

- During the lab sessions code fragments are written to create, destroy and manipulate data structures using dynamic memory. Students are also requested to divide a given functionality into functions and write their code (PO: c).

- During an eight-week period students are divided into teams and they must execute a project entailing the design of a software
application containing multiple milestones, deliverables and objectives(PO: d).

- Students are requested in several activites throughout the course to search for auxiliary documents to support the information studied in a topic. In their final report, they must acknowledge the information sources they used (PO: i).

- Use of the following tools: compiler, IDE, version
control and emulator in multiple laboratory sessions (PO: k).

During these activities the teaching staff reviews the student work in the class, supervises the lab sessions, answers questions in course forum, maintains at least one hour a week of office hours and calls for plenary office hours upon demand.

ASSESSMENT SYSTEM

Continuous evaluation (60% of the total):
- 1 midterm exam (20%)
- Project carried out during the course (40%)
Final exam (40%)
- A minimum of 40% of the final grade must be taken from the final exam

Extraordinary exam
Maximum grade between option 1 and option 2.
Option 1:
100% exam of the extraordinary call
Option 2:
60% continuous evaluation plus 40% extraordinary exam.
Students must take a minimum of 40% of the final grade of the exam of the extraordinary call.

% end-of-term-examination:	40
% of continuous assessment (assigments, laboratory, practicals…):	60

BASIC BIBLIOGRAPHY

- Steve Oualline: Practical C Programming, Proquest, 1991