uc3m | Universidad Carlos III de Madrid

Systems Architecture

Academic Year: (2019 / 2020) **Review date: 20/04/2020 10:02:38**

Department assigned to the subject: Telematic Engineering Department

Coordinating teacher: ESTEVEZ AYRES, IRIA MANUELA

Type: Compulsory ECTS Credits : 6.0

Year : 2 Semester : 1

REQUIREMENTS (SUBJECTS THAT ARE ASSUMED TO BE KNOWN)

Programming, Systems Programming and Digital electronics.

OBJECTIVES

1. The student must be able to design a software system containing non-trivial data structures, dynamic memory management, processses and interprocess communication techniques and using engineering techniques to translate a set of given high level constraints, derived from a hypothetical industrial setting, into a robust application.

2. The student must be able to understand the process and thread concepts, and understand different concurrency problems and apply the suitable synchronization mechanisms.

3. The student must be able to use proficiently the following industry-category tools: a compiler with different options to generate debugging information and to analyze the diagnostics produced while developing the application, a version controlled system to handle regular development flows, and profiling tools to analyze memory behavior in a software application.

4. The student must be able to: work effectively in a team to execute a project entailing the design of a software application on a mobile device, generate ideas collaboratively in a team to promote the exchange of information, organize the work in a team to optimize its performance and comply with the project requirements, and divide tasks effectively among the team members.

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

DESCRIPTION OF CONTENTS: PROGRAMME

1. Operating Systems
   - Processes and threads.
   - Concurrency mechanisms: locks, semaphores, monitors

2. The C programming language
   - Basic data types and flow constructions
   - Structure of a C application. The pre-processor, division in files, creating an executable.
   - Functions (with several parameters, variadic functions)
   - Pointer manipulation and pointers to functions.
   - File I/O

3. Dynamic memory management in C
   - Dynamic data structures
   - Memory leaks
   - Tools for detecting memory leaks

4. Multiprocess programming in C
   - Creation of processes.
   - Interprocess communication: signal, pipes.

5. Team project design

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 a mobile application, write code fragments to manipulate data structures, read/write fields, process data, creation of processes and threads, interprocess communication, 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 a nine-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).

- Write detailed meeting minutes with the action items and final conclusions, exchange information between teammates using chats, forums, email, explain the requirements derived from the specification of a work module, and the solution decided by the team (PO: g).

- 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: Virtual machines, compiler and version control 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

% end-of-term-examination/test:	45
% of continuous assessment (assigments, laboratory, practicals…):	55

The course assessment is performed with the following techniques:

***Individual assessment (up to 10 points): theory and lab

Theory assessment:

- Partial examination of the first block (20%)
- Final exam (45%). This exam has no minimum grade.

Lab assessment:

% end-of-term-examination/test:	45
% of continuous assessment (assigments, laboratory, practicals…):	55

- Lab examen of the first block (10%)
- Lab examen of the second block (10%)
- Lab exam of the Project (15%)

*** Optional points: Teamwork (up to 1 point).

Additionally, if and only if the student has passed the correspondent lab exam, a point can be added to the score.

This point is divided as follows:

- Version of the application to develop in the project in pairs (0,2)
- Final version of the application to develop in the project in teams (0,75)
- Presentation of the project (0.05)

The final score for the course is obtained as follows:

- If the individual assessment score is less than 4.2 points: Score = individual + teamwork
- If the individual assessment score is greater or equal to 4.2 points, then
  Score = max{individual+teamwork, 0.4*individual+ 6*teamwork}

Missed assessments will not be re-scheduled unless they are explicitly considered in the assessment regulations.

Due to the large percentage of the skills that directly depend on the objectives of the first year courses "Programming" and "Systems Programming", unless both of them have been pass, taking Systems Architecture is NOT recommended.

BASIC BIBLIOGRAPHY

- Abraham Silberschatz Operating system concepts, John Wiley & Sons, 2002

- C. Michael Pilato, Ben Collins-Sussman y Brian W. Fitzpatrick Version Control with Subversion, 2nd Edition, O'Reilly Media, Inc., 2008

- David Griffiths y Dawn Griffiths Head First C, O'Reilly Media, Inc., 2012

- Michael Kerrisk The Linux Programming Interface, No Starch Press, 2010

- Richard Reese Understanding and Using C Pointers, O'Reilly Media, Inc., 2013

- Stephen G. Kochan Programming in C, Fourth Edition, Addison-Wesley Professional, 2014

ADDITIONAL BIBLIOGRAPHY

- Brian W. Kernighan, Dennis M. Ritchie The C programming language, Prentice Hall, 1988

- Daniel Leuck, Patrick Niemeyer Learning Java, O'Reilly Media, Inc., 2013

- Félix García Carballeira et al. El lenguaje de programación C : diseño e implementación de programas, Pearson, 2001

- Jesús Carretero Perez, Felix García Carballeira et al. Problemas resueltos de programación en lenguaje C, Paraninfo, 2004

- Joost Visser; Sylvan Rigal, Gijs Wijnholds y Zeeger Lubsen Building Software Teams, O'Reilly Media, Inc., 2016

- Steve McConnell Code Complete, Second Edition, Microsoft Press, 2004