LICENCIATURA EN CIENCIA DE DATOS
Computación Concurrente
2020-I

Horario:	Martes & Jueves, 12:00 - 14:00	Viernes 13:00 - 14:00
Salón:	S-201	S-301

Instructor: Oscar A. Esquivel Flores
Cubículo: IIMAS-325
e-mail: oscar.esquivel@iimas.unam.mx

Objetivo General

Al finalizar el curso el alumno será capaz de realizar la construcción de un modelo de programación concurrente mediante el uso de distintas técnicas propias de la misma, como el paso de mensajes para resolver problemas de comunicación y sincronización entre procesos.

Pre-requisitos

- Se apreciará tener conocimientos básicos de algún lenguaje de programación de alto nivel como PYTHON y/o JULIA.
- Es deseable tener conocimiento de la terminal y línea de comandos.
- Es importante, aunque no obligatorio, tener conocimientos de herramientas básicas para la programación en ciencia de datos como repositorios, editores, *jupyter-notebooks*, sistema operativo LINUX o UNIX-like.
- No necesario pero deseable, el conocimiento de algún lenguaje del siglo pasado como JAVA, C, C++

Índice Temático del Curso

- Introducción
- Programación Secuencial
- Programación Concurrente
- Semáforos
- Monitores
- Paso de Mensajes
- Llamadas a procedimientos remotos
- Procesamiento Paralelo y Distribuido

Planificación del Curso

Fechas	Temas de clase
Ago 6, 8 & 9	Presentación Introducción al curso Práctica semanal
Ago 13, 15 & 16	Conceptos fundamentales Procesamiento concurrente,paralelo y distribuido Lenguajes para programación paralela y distribuida Introducción a Python para procesamiento paralelo Práctica semanal
Ago 20, 22 & 23	Programación secuencial Estructura, ventajas y limitaciones Ejemplos programación secuencial Práctica Semanal
Ago 27, 29 & 30	Programación concurrente Proceso concurrente y hebras de control Práctica semanal
Sep 3, 5 & 6	Multi-hebra y condiciones de competencia Sincronización Práctica Semanal
Sep 10, 12 & 13	Secciones críticas, exclusión, seguridad y eventual entrada Hebras de conrol Práctica semal
Sep 17, 19 & 20	Procesos concurrentes: productores y consumidores **Examen parcial**
Sep 24, 26 & 27	Semáforos Exclusión mutua Expresión implícita y explícita de semáforos Practica semanal
Oct 1, 3 & 4	Monitores Sincronización: monitores y semáforos Variables de condición y señalización Ejemplos Práctica semanal
Oct 8, 10 & 11	Paso de mensajes Canal, envío y recepción Práctica semanal
Oct 15, 17 & 18	Paso de mensajes síncrono, asíncrono y condicional Implementacion paso de mensajes Práctica semanal
Oct 20, 24 & 25	**Examen parcial** Revisón de examen Práctica semanal
Oct 29, 31 & Nov 1	Procedimientos remotos Sistema cliente-servidor Procedimiento remoto condicional Ejemplos e implementación Práctica semanal
Nov 5, 7 & 8	Procesamiento Paralelo y Distribuido Procesamiento síncrono, asíncrono Tipos de paralelismo Práctica semanal
Nov 12, 14 & 15	Ejemplo e implementación Práctica semanal **Examen parcial**
Nov 19, 21	Entrega de proyecto Evaluación final
Nov 26, 28	**Examen final**

Evaluación

El enfoque del curso es constructivista y se hará énfasis en la parte practica. Las actividades y porcentajes a evaluar son los siguientes:

- Examen de certificación (15 %)
- Exámenes parciales (50 %)
- Prácticas (P) - Tareas (T) (15 %)
- Proyecto final (20 %)

En caso de no aprobar el curso podrá presentarse un examen final (100 %) el cual podrá ser teórico y/o práctico.

Bibliografía

Básica:

- Bowman, H. (2010). Concurrency Theory: Calculi an Automata for Modelling Untimed and Timed Concurrent Systems. Springer: USA
- Forbes, E. (2017). Learning Concurrency in Python: Build highly efficient, robust, and concurrent applications, Birmingham: Packt
- Khot, A. (2018). Concurrent patterns and best practices. Birmingham: Packt.
- Lanaro G., Nguyen Q., Kasampalis S.(2019). Learning Path: Advanced Python Programming, Birmingham: Packt.
- Nguyen, Q. (2018). Mastering Concurrency in Python: Create faster programs using concurrency, asynchronous, multithreading, and parallel programming, Birmingham: Packt.
- Springer Berlin Heidelberg. (2013). Concurrent Programming: Algorithms, Principles, and Foundations. Berlin, Heidelberg.
- Summerfield M. (2013). Python in Practice: Create Better Programs Using Concurrency, Libraries, and Patterns. Addison-Wesley.

Complementaria y referencias

- Argonne National Laboratory. Designing and Building Parallel Programs. Chicago, Ill.
- Balaji, P. Programming models for parallel computing. MIT: USA
- Dooley, J. (2017). Software Development, Design and Coding. Berkeley, CA: Apress.
- Lenguaje de programación Julia: https://julialang.org/.
- Lenguaje de programación Python: https://www.python.org/
- Lutz M. (2013). Learning Python, O'Reilly.
- Mattson, T., Sanders, B., & Massingill, B. (2010). Patterns for parallel programming. Boston: Addison-Wesley.
- Ramalho L. (2015). Fluent Python, O'Reilly.