FCEFyN
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS y NATURALES

UNC Universidad Nacional de Córdoba

Asignatura: **Programación Concurrente y Paralela**

Código:10-09806	RTF	8
Semestre: Sexto	Carga Horaria	96
Bloque: Tecnologías Básicas	Horas de Práctica	45

Departamento: Computación

Correlativas:

- Programación Avanzada

Contenido Sintético:

- Visión e Historia de los sistemas concurrentes.
- Elementos, soporte y herramientas relevantes de la programación concurrente.
- Paradigmas, algoritmos y patrones de programación concurrente.
- Aspectos formales y especificación de concurrencia, paralelismo, sincronización, eventos y conflictos en sistemas embebidos y reactivos.
- Gestión del tamaño y complejidad de los modelos.
- Modelado de sistemas para evaluar el rendimiento de la dinámica de sistemas concurrentes.
- Nociones de programación paralela: estructura paralela de un algoritmo, coordinación de procesos paralelos, speedup y escalabilidad, ejemplos en sistemas multihilo.
- Aplicación práctica de integración de hardware software de sistemas.

Competencias Genéricas:: (Contribución: A = Alta; M = Media; B = Baja)

- CG1: Identificar, formular y resolver problemas de ingeniería. (A)
- CG4: Utilizar de manera efectiva las técnicas y herramientas de aplicación en ingeniería. (A)

Aprobado por HCD: 1042-HCD-2023 RES: Fecha: 27/11/2023

Competencias Específicas: (Contribuciones: A = Alto; M = Medio; B = Bajo)

- CE6.1: Conocer y poder diseñar la estructura de sistemas operativos, la administración de procesos, la gestión de memoria, la administración de archivos, la gestión de dispositivos de entrada/salida y la implementación de políticas de seguridad. (A)
- CE6.2: Analizar, conocer, interpretar y aplicar mecanismos de comunicación y sincronización. (A)
- CE6.3: Proyectar, desarrollar, dirigir, controlar, construir, operar y mantener sistemas operativos embebidos, para dispositivos móviles y de tiempo real. (M)
- CE7.2.2: Sintetizar, diseñar, desarrollar y analizar programas lenguajes de programación de bajo nivel, como C y C++. (A)
- CE7.3.1: Sintetizar, diseñar, simular, construir y analizar circuitos y sistemas de control en tiempo continuo y tiempo discreto, aplicables a cualquier área del alcance de la profesión.

Presentación

La computación concurrente y paralela es de gran relevancia en la actualidad debido a que los sistemas informáticos modernos disponen de procesadores multinúcleo, los cuales pueden ejecutar tareas heterogéneas simultáneamente. Estos sistemas multinúcleo, requieren técnicas formales de diseño y modelado para su desarrollo e implementación. Ejemplos de estos sistemas son los sistemas ciber físicos (CF), reactivos (RS), embebidos, críticos y dirigidos por eventos los cuales interactúan con variables y eventos externos heterogéneos y no determinísticos.

En este contexto la programación concurrente es fundamental en educación informática y requiere abordar comunicación, coordinación, modelos de memoria compartida, atomicidad, sincronización y espera condicional para garantizar eficacia y eficiencia en el diseño y desarrollo de este tipo de sistemas.

Programación concurrente y paralela es una asignatura que aborda la comprensión y resolución de las problemáticas informáticas inherentes a las aplicaciones con ejecución de varios subprocesos simultáneos o tareas en paralelo para aprovechar al máximo los recursos. De esta manera es posible mejorar el rendimiento, acelerar la ejecución de los programas y resolver la problemática del uso de los recursos informáticos disponibles. Esta asignatura pertenece al tercer año (sexto semestre) de la carrera Ingeniería en Computación y en esta instancia de cursado se espera que el estudiante haya adquirido competencias en programación avanzada.

Las líneas conceptuales abordadas en esta asignatura permiten inferir que un estudiante que certifique la promoción de la asignatura habrá desarrollado habilidades para: identificar situaciones donde el uso compartido de recursos tiene potencial impacto en el funcionamiento del sistema y diseñar una solución a través de la gestión, la creación y el manejo de subrutinas o hilos de ejecución (threads), implementar la sincronización de subprocesos, crear y administrar ejecutores (thread pool executors), controlar el flujo de su aplicación mediante Fork / Join framework, analizar el paralelismo y explicar los factores que limitan la aceleración alcanzable, utilizar estructuras de datos para programas concurrentes, adaptar el comportamiento predeterminado de algunas clases de concurrencia a sus necesidades y diseñar, verificar y validar casos de aplicación de concurrencia con el lenguaje Java.

Contenidos

Capítulo 1: Introducción a la programación concurrente y paralela

Introducción y visión histórica de los sistemas concurrentes. Concurrencia y paralelismo. Testing, simulación y sus limitaciones. El problema de la corrección del software y ejemplos de fallos.

Capítulo 2: Elementos, soporte y herramientas relevantes de la programación concurrente.

Conceptos y definiciones de Sistemas Críticos y reactivos. Interleaving. Problemas comunes de los programas concurrentes. Ejecución de programas concurrentes. El enfoque de modelar. Diagramas UML.

Capítulo 3: Descripción y control de procesos
Definición de un proceso y de una subrutina o thread. Descripción de un proceso. Estados de un proceso y estados de un thread. Control y gestión de threads.

Capítulo 4: Autómatas finitos y Lenguajes Regulares
Introducción e historia de los lenguajes y gramáticas formales. Teoría de autómatas. Símbolo, vocabulario, cadena, lenguaje gramática y autómata. Definición formal de gramática.

Capítulo 5. Jerarquía de las gramáticas y Máquina de Turing.
Jerarquía de las gramáticas. Expresiones regulares. Máquinas de Mealy y Moore. Máquina de Turing y equivalencias. Autómatas finitos deterministas y no deterministas.

Capítulo 6. Concurrencia: exclusión mutua y sincronización.
Principios generales del paradigma de la concurrencia. Conceptos y definiciones. Procesos concurrentes y memoria compartida. Herramientas de sincronización. Exclusión mutua. Sección crítica y condición de carrera.

Capítulo 7. Sincronización, eventos y conflictos en sistemas embebidos y reactivos.
Conceptos y definiciones. Primitivas de sincronización de hilos (threads). Semáforos. Monitores. Paso de mensajes. Ejecutores y pool de hilos Problema de los lectores/escritores. Problema de la cena de los filósofos. Problema de la barbería y otros.

Capítulo 8. El enfoque de modelar: Redes de Petri
Redes de Petri ordinarias. Matriz de incidencia. Ecuación general de estado. Propiedades de las Redes de Petri (vivacidad, interbloqueo, etc). Modelo de proceso concurrente. Invariante de plaza y de transición. Grafo de alcanzabilidad y cobertura. Herramientas para el análisis de sistemas modelados con Redes de Petri.

Capítulo 9. Redes de Petri extendidas. Evaluación de rendimiento.
Redes de Petri temporales y formalismos. Semántica de las Redes de Petri temporales. Propiedades. Grado de Sensibilizado y prioridad. Semántica de tiempo débil y fuerte. Modelado de sistemas de tiempo real mediante Redes de Petri con tiempo. Redes de Petri coloreadas y su aplicación.

Capítulo 10. Integración
Análisis y desarrollo de un sistema concurrente y paralelo, caso de aplicación. Descripción de propiedades. Diseño y determinación de los threads necesarios y máximos en ejecución para la aplicación. Análisis de rendimiento. Validación formal de operatividad. Aplicación práctica de integración de hardware software de sistemas.

Metodología de enseñanza

Dentro del cuerpo de conocimiento dedicado a la pedagogía, se establecen fundamentales criterios didácticos y estrategias de enseñanza. En el contexto de la presente asignatura y en relación a estos criterios, la premisa fundamental es la implementación de una metodología educativa centrada en el estudiante, con un enfoque decidido en el desarrollo de competencias. Esta perspectiva pedagógica se sustenta en los principios del constructivismo. De esta manera y siguiendo el modelo de enseñanza por explicación y contrastación de casos para que el estudiante pueda relacionar el conocimiento cotidiano con los conocimientos de ingeniería se alcanza el progreso en el aprendizaje. A su vez en los prácticos se aplica un enfoque activo-participativo donde se fomenta la resolución de problemas, el trabajo en equipo y el pensamiento crítico frente a problemas de diseño para alcanzar el desarrollo de habilidades y la toma de decisiones en grupo.

En esta asignatura el aprendizaje basado en proyectos desempeña un rol esencial, debido a que los estudiantes a lo largo de la asignatura deben resolver trabajos prácticos en los cuales deben aplicar los conceptos aprendidos y las técnicas de manera recíproca a lo que sucedería en la vida real. Enfrentando decisiones de diseño, trade-offs o relaciones relaciones de compromiso donde es necesario poder aplicar las competencias aprendidas para argumentar las decisiones de diseño tomadas frente al grupo de pares y frente al docente.

Las nuevas tecnologías se utilizan activamente, con aplicación de casos y contrastación de ejemplos contra Large Language Models LLM AI (Chat GPT). Se actualiza constantemente material audio visual, haciendo uso de las herramientas y el conocimiento obtenido en cursos de mejora continua. A través del campus virtual LEV (Moodle), se establece un canal de contacto bidireccional con amplio material didáctico que en conjunto con las plataformas de mensajería brindan al estudiante la posibilidad de estar en contacto directo con el docente.

Evaluación

Dentro del marco de esta propuesta teórico-práctica, esta cátedra ha optado por un proceso de evaluación mixto que incluye dos (2) exámenes parciales con oportunidad de recuperación, desarrollo de dos (2) trabajos prácticos de laboratorio y examen coloquio final integrador.

Los exámenes parciales están diseñados para evidenciar tanto el conocimiento como las competencias adquiridas y sus contenidos implican tanto conceptos teóricos como prácticos. Los trabajos prácticos de laboratorio representan el verdadero desafío de aplicación de las competencias adquiridas como un todo. Los estudiantes trabajan en equipo para diseñar, desarrollar, implementar y documentar soluciones a sistemas concurrentes y paralelos. Los proyectos incluyen el análisis de las características comportamentales del sistema, comprender la interacción de la aplicación, poner especial atención en la detección y prevención de situaciones problemáticas relacionadas con la concurrencia, el paralelismo y los recursos compartidos en el sistema así como la decisión de las técnicas utilizadas para resolverlos.

Estos trabajos tienen instancia de entrega y de defensa grupal, en la cual los estudiantes deben articular las competencias técnicas específicas adquiridas y las competencias genéricas, para argumentar y validar el desarrollo propuesto.
Estas entregas se evalúan individualmente con los siguientes criterios prefijados:

- Calidad del diseño
- Organización
- Estructura de la solución
- Calidad de la implementación (código)
- Escritura académica y documentación
- Claridad en la defensa-exposición.

Esta modalidad fomenta la habilidad comunicacional y la capacidad de argumentación de decisiones de diseño frente a terceros.
En la instancia final el estudiante debe realizar un coloquio integrador en el cual mediante una exposición dialogada con el docente, se recorren los conocimientos adquiridos y analizan las soluciones implementadas en los trabajos prácticos de laboratorios.

Si bien cada trabajo puede favorecer el desarrollo de una determinada competencia en particular y es de esperar la evidencia de esto hacia la conclusión de dicha actividad, la evaluación será contínua a lo largo de todas las actividades propuestas.

Si bien es posible inferir que cada actividad propuesta puede impactar sobre el desarrollo de competencias particulares ya sean específicas o genéricas y se anticipa que esto será evidente al finalizar cada actividad; es importante destacar que la evaluación se llevará a cabo de manera continua a lo largo de todas las actividades propuestas.
Las actividades propuestas están diseñadas para que su conclusión sea indicativo del desarrollo de los resultados de aprendizaje propuestos.

Condiciones de aprobación

Condición de regularización:
Para alcanzar esta condición el estudiante debe cumplir con:

- Asistencia y participación a clases mayor o igual al 80%.
- Aprobación de ambos exámenes Teórico-Prácticos con porcentaje mayor o igual a 60%.
- Aprobación de un trabajo práctico de laboratorio.

Condición de promoción:
Para alcanzar esta condición el estudiante debe cumplir con:

- Todos los requisitos indicados en la condición de alumno regular.
- Debe aprobar ambos trabajos prácticos de laboratorio.
- Debe aprobar el coloquio final integrador.

Actividades prácticas y de laboratorio

Además de las actividades indicadas en el punto Evaluación, en las clases prácticas se desarrollan actividades en conjunto con el dictado de clases resolviendo problemas de

acuerdo a los contenidos que se van desarrollando entre los cuales se destacan las siguientes actividades:

- Desarrollo de diagramas UML - Casos de aplicación
- Creación de hilos (Threads) y su interacción. Debug.
- Control de un thread (Interrupt, sleep, join)
- Herramientas de sincronización (Synchronized, locks)
- Herramientas avanzadas de sincronización (Read_write_locks, fairness, semáforos)
- Herramientas avanzadas de concurrencia (Uncont_exceptions, daemon_threads, local_thread_variables, count_down_latch)
- Herramientas avanzadas de sincronización en java (Cyclic_barrier, phaser, exchanger)
- Control y gestión de subrutinas concurrentes y paralelas (Factory, thread_factory, executors)

Resultados de aprendizaje

Los resultados de aprendizaje esperados de la asignatura son:

1. Analizar e interpretar correctamente el dominio de un problema
2. Evaluar las características y requerimientos de un sistema concurrente y paralelo de mediana dimensión.
3. Identificar y comprender situaciones donde el uso compartido de recursos tiene potencial impacto en el funcionamiento del sistema
4. Analizar el paralelismo inherente a un simple algoritmo secuencial.
5. Seleccionar métodos apropiados para medir el rendimiento de algoritmos multiproceso.
6. Explicar y calcular los tiempos de respuesta de un sistema concurrente y paralelo y explicar los factores que limitan la aceleración alcanzable.
7. Explicar las limitaciones de la escalabilidad.
8. Reconocer el potencial del co-diseño hardware-software en circunstancias de complejidad modesta.
9. Gestionar la creación y el manejo de subrutinas o hilos de ejecución (threads),
10. Identificar, diseñar e implementar la sincronización de subprocesos reconociendo condiciones de carrera y exclusión mutua.
11. Gestionar un conjunto de threads, crear y administrar ejecutores (thread pool executors)
12. Crear un plan de prueba y generar casos de prueba para un sistema basado en computadora de complejidad media, seleccionando una combinación apropiada de cantidad y niveles de pruebas para garantizar la calidad del sistema.
13. Adaptar el comportamiento predeterminado de algunas clases de concurrencia a sus necesidades y diseñar, verificar y validar casos de aplicación de concurrencia con el lenguaje Java.
14. Realizar, como parte de una actividad de equipo, una inspección de un diseño de sistema informático de tamaño mediano.

Estos resultados de aprendizaje cubren las competencias genéricas y específicas y serán evaluados en las instancias de evaluación previstas mediante rúbricas elaboradas a tal efecto.

Bibliografía

- J. T. P. Méndez, Programación concurrente: Editorial Paraninfo, 2003.
- J. F. González, Java 9 Concurrency Cookbook, Second Edition, Packt Publishing Ltd, 2017.
- D. Lea, Concurrent Programming in Java™: Design Principles and Patterns, Second Edition, 1999.
- W. Stallings, Sistemas operativos vol. 2: Prentice Hall, 2004.
- M. Diaz, Petri Nets Fundamental Models, Verification and Applications. NJ USA: John Wiley & Sons, Inc, 2009.
- R. David and H. Alla, Discrete, continuous, and hybrid Petri nets. Springer Science & Business Media, 2010.