PROGRAMACIÓN CONCURRENTE

Año 2018

Carrera/ Plan:
Licenciatura en Informática Plan 2015-2012-2007
Licenciatura en Sistemas Plan 2015-2012-2007
Analista Programador Universitario Plan 2015-2007

Año: 3°
Régimen de Cursada: *Semestral* (2do semestre)
Carácter (Obligatoria/Optativa): Obligatoria
Correlativas: Introducción a los Sistemas Operativos- Seminario de Lenguajes- Taller de Lecto-comprensión y Traducción en Inglés
Profesor/es: Marcelo Naiouf, Franco Chichizola, Laura De Giusti, Enzo Rucci
Hs. semanales : 6 hs

FUNDAMENTACIÓN

La temática de la Concurrencia es central en el desarrollo de la Ciencia Informática, en particular por el creciente desarrollo de arquitecturas multiprocesador que permiten implementar físicamente los conceptos teóricos de concurrencia "real".
El impacto de la concurrencia se refleja en diferentes ámbitos de la disciplina tales como las arquitecturas, los sistemas operativos, los lenguajes y el diseño y desarrollo de aplicaciones. En este sentido, se impone que los futuros profesionales sean capaces de desarrollar soluciones que utilicen adecuadamente la tecnología disponible con fundamentos teóricos firmes.

OBJETIVOS GENERALES

Brindar los conceptos fundamentales de Concurrencia en software. Analizar la semántica y sintaxis para especificar concurrencia. Analizar el concepto de sistema concurrente compuesto por la arquitectura, el sistema operativo y los algoritmos. Estudiar la sincronización de procesos concurrentes por memoria compartida y mensajes. Vincular la concurrencia en software con los conceptos de procesamiento distribuido y paralelo. Desarrollar estudios de casos con diferentes lenguajes/ herramientas para concurrencia.

CONTENIDOS MINIMOS

- Especificación de la ejecución concurrente.
- Comunicación y sincronización.
- Concurrencia con variables compartidas.
- Concurrencia con pasajes de mensajes.
- Lenguajes de programación concurrente.
- Introducción a los conceptos de procesamiento paralelo.

PROGRAMA ANALÍTICO

1. Conceptos básicos

Objetivos de los sistemas concurrentes.
Procesamiento secuencial, concurrente y paralelo. Características.
Evolución histórica. El modelo de CSP (Communicating Sequential Processes)
Procesos. Programa concurrente. No determinismo.
Clases de aplicaciones. Multithreading, Cómputo paralelo y distribuido. Concurrencia y paralelismo.
Algoritmos concurrentes, distribuidos y paralelos.
Areas de estudio en sistemas concurrentes.
Relación con la arquitectura. Monoprocesadores. Multiprocesadores: Procesadores híbridos.
Clasificaciones y ejemplos. Tendencias actuales en procesadores.
Conceptos de arquitecturas Grid y Cloud. Memoria compartida distribuida.
Relación con el sistema operativo. Requerimientos para el sistema operativo.
Relación con el lenguaje. Requerimientos para el lenguaje.
Sincronización y comunicación. Sincronización por exclusión mutua y por condición. Comunicación por memoria compartida y por mensajes.
Prioridad, granularidad, deadlock, manejo de recursos.
Paradigmas de resolución de programas concurrentes: iterativo, recursivo o *divide & conquer*, *pipeline* o productor consumidor, cliente/servidor y sus variantes, *peers* o interacción por pares.

2. *Concurrencia y sincronización*

Aspectos de programación secuencial
Especificación y semántica de la ejecución concurrente. La sentencia *co* y *process*
Acciones atómicas y sincronización.
El problema de interferencia. Historias válidas e inválidas.
Atomicidad de grano fino y de grano grueso.
La propiedad de “A lo sumo una vez”.
La sentencia Await. Semántica.
Técnicas para evitar interferencia.
Propiedades de seguridad y vida.
Políticas de scheduling y Fairness.
Requerimientos para los lenguajes de programación.
Problemas en sistemas distribuidos.

3. *Concurrencia con variables compartidas*

Sincronización por variables compartidas

Sincronización de grano fino.
Secciones críticas (SC). Definición del problema. Propiedades necesarias de las soluciones.
Soluciones de tipo spin-locks al problema de la SC.
Algoritmos clásicos de soluciones fair al problema de la SC (tie-breaker, ticket, bakery).
Implementación de sentencias Await arbitrarias.
Sincronización barrier. Definición. Planteo de soluciones (contador compartido, flags y coordinadores, árboles, barreras simétricas, butterfly).
Algoritmos *data parallel*. Ejemplo: Computación de prefijos.

Sincronización por semáforos
Semáforos. Sintaxis y semántica.
Usos básicos y técnicas de programación.
Soluciones a SC y barreras.
Semáforos binarios divididos (*split*).
Exclusión mutua selectiva.
La técnica "*passing the baton"*. Definición y aplicaciones.
Sincronización por condición general.
Alocación de recursos. Políticas de alocación. SJN.
Ejemplos clásicos: filósofos, lectores/escritores, productores/consumidores con buffer limitado, etc.
Semáforos en lenguajes reales: Pthreads. Ejemplos.

Sincronización por monitores
Evolución histórica a partir de semáforos. Conceptos de regiones críticas condicionales.
Monitores. Sintaxis y semántica.
Sincronización en monitores.
Disciplinas de señalización: "Signal and wait" y "Signal and continue": diferencias y efecto sobre la programación.
La técnica "*passing the condition".*
Ejemplos clásicos: buffer limitado, lectores y escritores, alocación, etc.
Diseño de un reloj lógico. Alternativas.
El problema del peluquero: rendezvous.
Scheduling de discos. Ejemplo. Enfoques alternativos para la sincronización.
Monitores en lenguajes reales: Java, Pthreads. Ejemplos.

Implementaciones
Conceptos de implementación de procesos en arquitecturas mono y multiprocesador.
Kernel monoprocesador y multiprocesador.

4. Programación distribuida. Concurrencia con pasaje de mensajes

Programas distribuidos. Definición.
Relación entre mecanismos de comunicación.
Clases básicas de procesos: productores y consumidores, clientes y servidores, peers.
Control de concurrencia en Sistemas Distribuidos.

Mensajes asincrónicos
Sintaxis y semántica. Canales. Operaciones.
Filtros. Redes de Filtros.
Clientes/Servidores. Algoritmos clásicos. Monitores activos. Continuidad conversacional.
Peers. Intercambio de valores. Cálculo de la topología de una red.
Mensajes asincrónicos en lenguajes reales MPI. Extensión de lenguajes secuenciales con bibliotecas específicas.
Ejemplos.
El concepto de *bag of tasks*. Espacio de tuplas e interacción entre procesos. Estructuras de datos distribuidas. Ejemplos

Mensajes sincrónicos
Sintaxis y semántica.
Conceptos de CSP.
Comunicación guardada. Sintaxis y semántica.
Filtros. Clientes y servidores. Asignación de recursos.
Interacción entre procesos paralelos. Ejemplos.
Mensajes sincrónicos en lenguajes reales.
Ejemplos

Remote Procedure Calls y Rendezvous.
Sintaxis y semántica. Similitudes y diferencias.
RPC: Sincronización en módulos.
Discusión revisada de aplicaciones.
RPC en lenguajes reales: Java. RMI. Ejemplos.
Rendezvous en lenguajes reales: Ada. Tasks y sincronización. Ejemplos

Primitivas múltiples y otros enfoques
Sintaxis y semántica.
MPD. Componentes de programa. Comunicación y sincronización. Ejemplos
Conceptos básicos sobre otros lenguajes/plataformas

Implementaciones
Conceptos de implementación de mecanismos de comunicación y sincronización en ambientes distribuidos.

5. *Paradigmas de interacción entre procesos distribuidos*
Resolución de problemas mediante diferentes paradigmas de interacción entre procesos: Servidores replicados, heartbeat, pipeline, prueba-eco, broadcast, token passing, manager/workers. Ejemplos. Comparación de alternativas.

6. *Introducción a la Programación Paralela*
Objetivos del cómputo paralelo. Necesidad del paralelismo. Areas de aplicación.
Computación científica.
Diseño de algoritmos paralelos.
Métricas. Conceptos de speedup y eficiencia.
La ley de Amdahl y la ley de Gustafson.
Concepto de escalabilidad.

En la práctica se plantean ejercicios que pueden ser realizados como trabajo experimental sobre arquitecturas multiprocesador distribuidas (clusters), multiprocesadores con memoria compartida e híbridos.

BIBLIOGRAFÍA

- Andrews G. “Foundations of Multithreaded, Parallel and Distributed Programming”, Addison Wesley, 2000
- Barnes J., “Programming in Ada 2005 with CD”, Addison Wesley, 2006.
- Ben-Ari, M. “Principles of Concurrent and Distributed Programming, 2/E”. Addison-Wesley. 2006. ISBN 0-321-31283-X
- Chandy, Misra, “Parallel Program Design. A Foundation”, Addison Wesley, 1988.
- Downey, Allen. “The Little Book of Semaphores, Second Edition”. Free book disponible en http://www.freetechbooks.com/the-little-book-of-semaphores-second-edition-t519.html, 2007.
- Filminas de las clases teóricas.
- Grama A., Gupta A., Karypis G., Kumar V., "An Introduction to Parallel Computing. Design and Analysis of Algorithms", Pearson Addison Wesley, 2nd Edition, 2003
- Herlihy M., Shavit N., “The Art of Multiprocessor Programming”. Morgan Kaufmann. Revised reprint, 2012.
- Hoare C., “Communicating Sequential Processes”, Englewood Cliffs, Prentice Hall, 1985
- Brinch Hansen, P., “Studies in Computational Science. Parallel Programming Paradigms”, Prentice Hall, 1995.
- Jordan H.F., Alaghband G., Jordan H.E., "Fundamentals of Parallel Computing", Prentice Hall, 2002
- Naiouf, De Giusti A., De Giusti L, Chichizola, “Conceptos de concurrencia y paralelismo”, UNLP, a publicar 2018.
- Pacheco, P. “An introduction to parallel programming”. Morgan Kaufmann, 2011.
- Raynal M. “Concurrent Programming: Algorithms, Principles, and Foundations”. Springer, 2012.
- Taubenfeld, Gadi. “Synchronization Algorithms and Concurrent Programming”. Prentice Hall. 2006.

METODOLOGÍA DE ENSEÑANZA

La actividad curricular se organiza en:

- clases teóricas, a cargo de un profesor (en dos horarios, de mañana y de tarde), en las cuales se imparten los temas de la asignatura. Los alumnos podrán asistir a cualquiera de los dos horarios.
- explicaciones de práctica, a cargo de un profesor y/o JTP, y que actúan a modo de articulación entre teoría y práctica y donde se plantean y resuelven problemas “tipo”.
- clases prácticas, a cargo de auxiliares docentes (ayudantes coordinados por JTP, en horario de mañana, tarde y sábados a la mañana), donde los alumnos trabajan sobre los ejercicios propuestos en la guía de trabajos prácticos. Los alumnos pueden asistir indistintamente a cualquiera de los horarios (o a los tres).
- clases de consulta (de teoría y práctica).
- periódicamente se publican cuestionarios de teoría a modo de guía a fin de que los alumnos reflexionen sobre los puntos más importantes.

El reglamento y cronograma tentativo son conocidos por los alumnos desde el inicio de la actividad curricular.

Se utiliza un entorno virtual de enseñanza-aprendizaje (IDEAS), donde se encuentran disponibles clases, guías de TP, avisos, resultados de exámenes, etc.

Para las clases teóricas y las explicaciones de práctica se utiliza PC, cañón y pizarrón.

Los alumnos pueden realizar prácticas en PC usando distintos lenguajes/bibliotecas que soportan concurrencia.

EVALUACIÓN

La cursada puede obtenerse de dos maneras:
a) Aprobar un parcial normal en alguna de las 3 fechas que se dispondrán (parcial y 2 recuperatorios)
b) Alternativamente y de tipo opcional, se cuenta con un régimen de promoción con las siguientes características:
- se tomarán durante el curso 3 parcialitos prácticos.
- si la nota de los 3 parcialitos es mayor o igual que 8, el alumno obtiene la cursada.
- si la nota promedio de los parcialitos es mayor o igual que 6 y el alumno aprobó al menos 2 de los 3 parcialitos (nota mayor o igual que 6), podrá obtener la cursada aprobando un parcial reducido que se rinde en las mismas fechas que el normal.
Se considerarán en condición de "Desaprobado" los alumnos que rindan y obtengan "D" al menos en 2 de las 3 fechas de evaluación parcial.

El final puede aprobarse de dos maneras:
a) Final tradicional (teórico-práctico) en mesa de finales.
b) Alternativamente y de tipo opcional, cumpliendo las siguientes condiciones (cada una tiene como precondición cumplir con la anterior):

i) Rendir al menos 2 de 3 parcialitos teóricos que se tomarán en las clases de la mañana.
ii) Luego de aprobar la cursada, rendir y aprobar un parcial teórico.
iii) Cumplido ii):
- si la nota del parcial teórico es >= 7 el alumno queda habilitado para rendir un coloquio en mesa de final dentro del semestre siguiente.
- si la nota del parcial teórico es >=4 y <7, se asigna un trabajo individual. El mismo debe ser desarrollado y defendido en un coloquio en fecha de final dentro del semestre siguiente.

En caso de presentarse a rendir final tradicional, la promoción de teoría caduca.

Si bien se toman en los mismos días, los parcialitos de práctica y de teoría son independientes (el alumno puede rendir uno de los dos o ambos).

CRONOGRAMA DE CLASES Y EVALUACIONES

El comienzo de clases está previsto para la semana del 13 de agosto.
El cronograma detallado se pone en conocimiento de los alumnos al inicio del curso.
El esquema preliminar por bloque de las clases y evaluaciones es el siguiente:

Clase	Fecha	Contenidos/Actividades
1 y 2	13 y 20/8	Conceptos básicos. Comunicación y sincronización. Interferencia.
3 a 6	27/8 al 17/9	Concurrencia con memoria compartida.
7 a 9	24/9 al 15/10	Concurrencia con memoria distribuida
10	22/10	Introducción a la Programación Paralela.

Evaluaciones previstas	Fecha
1er parcialito	Semana del 17/9
2do parcialito	Semana del 15/10
3er parcialito	Semana del 29/10
Parcial	Semana del 5/11 o 12/11
1er recuperatorio	Semana del 19/11 o 26/11
2do recuperatorio	Semana del 3/12 o 10/12
Parcial teórico	Semana del 18/2/2019

Contacto de la cátedra (mail, página, plataforma virtual de gestión de cursos):

Plataforma virtual: ideas.info.unlp.edu.ar
Web: weblidi.info.unlp.edu.ar/catedras/concurrente/WEB/
Mail: mnaiouf@lidi.info.unlp.edu.ar, ldgiusti@lidi.info.unlp.edu.ar, francoch@lidi.info.unlp.edu.ar, erucci@lidi.info.unlp.edu.ar

Firma del/los profesor/es

PROGRAMACIÓN CONCURRENTE
(Redictado)

Año 2018

Carrera/ Plan:
Licenciatura en Informática Plan 2015-2012-2007
Licenciatura en Sistemas Plan 2015-2012-2007
Analista Programador Universitario Plan 2015-2007

Año: 3°
Régimen de Cursada: *Semestral* (1er semestre)
Carácter (Obligatoria/Optativa): Obligatoria
Correlativas: Introducción a los Sistemas Operativos- Seminario de Lenguajes- Taller de Lecto-comprensión y Traducción en Inglés
Profesor/es: Marcelo Naiouf, Franco Chichizola
Hs. semanales: 6 hs

FUNDAMENTACIÓN

La temática de la Concurrencia es central en el desarrollo de la Ciencia Informática, en particular por el creciente desarrollo de arquitecturas multiprocesador que permiten implementar físicamente los conceptos teóricos de concurrencia “real”.
El impacto de la concurrencia se refleja en diferentes ámbitos de la disciplina tales como las arquitecturas, los sistemas operativos, los lenguajes y el diseño y desarrollo de aplicaciones. En este sentido, se impone que los futuros profesionales sean capaces de desarrollar soluciones que utilicen adecuadamente la tecnología disponible con fundamentos teóricos firmes.

OBJETIVOS GENERALES

Brindar los conceptos fundamentales de Concurrencia en software. Analizar la semántica y sintaxis para especificar concurrencia. Analizar el concepto de sistema concurrente compuesto por la arquitectura, el sistema operativo y los algoritmos. Estudiar la sincronización de procesos concurrentes por memoria compartida y mensajes. Vincular la concurrencia en software con los conceptos de procesamiento distribuido y paralelo. Desarrollar estudios de casos con diferentes lenguajes/ herramientas para concurrencia.

CONTENIDOS MINIMOS

- Especificación de la ejecución concurrente.
- Comunicación y sincronización.
- Concurrencia con variables compartidas.
- Concurrencia con pasajes de mensajes.
- Lenguajes de programación concurrente.
- Introducción a los conceptos de procesamiento paralelo.

PROGRAMA ANALÍTICO

1. Conceptos básicos

Objetivos de los sistemas concurrentes.
Procesamiento secuencial, concurrente y paralelo. Características.
Evolución histórica. El modelo de CSP (Communicating Sequential Processes)
Procesos. Programa concurrente. No determinismo.
Clases de aplicaciones. Multithreading, Cómputo paralelo y distribuido. Concurrencia y paralelismo.
Algoritmos concurrentes, distribuidos y paralelos.
Areas de estudio en sistemas concurrentes.
Relación con la arquitectura. Monoprocesadores. Multiprocesadores: Procesadores híbridos.
Clasificaciones y ejemplos. Tendencias actuales en procesadores.
Conceptos de arquitecturas Grid y Cloud. Memoria compartida distribuida.
Relación con el sistema operativo. Requerimientos para el sistema operativo.
Relación con el lenguaje. Requerimientos para el lenguaje.
Sincronización y comunicación. Sincronización por exclusión mutua y por condición. Comunicación por memoria compartida y por mensajes.
Prioridad, granularidad, deadlock, manejo de recursos.
Paradigmas de resolución de programas concurrentes: iterativo, recursivo o *divide & conquer*, *pipeline* o productor consumidor, cliente/servidor y sus variantes, *peers* o interacción por pares.

2. *Concurrencia y sincronización*

Aspectos de programación secuencial
Especificación y semántica de la ejecución concurrente. La sentencia *co* y *process*
Acciones atómicas y sincronización.
El problema de interferencia. Historias válidas e inválidas.
Atomicidad de grano fino y de grano grueso.
La propiedad de “A lo sumo una vez”.
La sentencia Await. Semántica.
Técnicas para evitar interferencia.
Propiedades de seguridad y vida.
Políticas de scheduling y Fairness.
Requerimientos para los lenguajes de programación.
Problemas en sistemas distribuidos.

3. *Concurrencia con variables compartidas*

Sincronización por variables compartidas

Sincronización de grano fino.
Secciones críticas (SC). Definición del problema. Propiedades necesarias de las soluciones.
Soluciones de tipo spin-locks al problema de la SC.
Algoritmos clásicos de soluciones fair al problema de la SC (tie-breaker, ticket, bakery).
Implementación de sentencias Await arbitrarias.
Sincronización barrier. Definición. Planteo de soluciones (contador compartido, flags y coordinadores, árboles, barreras simétricas, butterfly).
Algoritmos *data parallel*. Ejemplo: Computación de prefijos.

Sincronización por semáforos
Semáforos. Sintaxis y semántica.
Usos básicos y técnicas de programación.
Soluciones a SC y barreras.
Semáforos binarios divididos (*split*).
Exclusión mutua selectiva.
La técnica "*passing the baton"*. Definición y aplicaciones.
Sincronización por condición general.
Alocación de recursos. Políticas de alocación. SJN.
Ejemplos clásicos: filósofos, lectores/escritores, productores/consumidores con buffer limitado, etc.
Semáforos en lenguajes reales: Pthreads. Ejemplos.

Sincronización por monitores
Evolución histórica a partir de semáforos. Conceptos de regiones críticas condicionales.
Monitores. Sintaxis y semántica.
Sincronización en monitores.
Disciplinas de señalización: "Signal and wait" y "Signal and continue": diferencias y efecto sobre la programación.
La técnica "*passing the condition".*
Ejemplos clásicos: buffer limitado, lectores y escritores, alocación, etc.
Diseño de un reloj lógico. Alternativas.
El problema del peluquero: rendezvous.
Scheduling de discos. Ejemplo. Enfoques alternativos para la sincronización.
Monitores en lenguajes reales: Java, Pthreads. Ejemplos.

Implementaciones
Conceptos de implementación de procesos en arquitecturas mono y multiprocesador.
Kernel monoprocesador y multiprocesador.

4. Programación distribuida. Concurrencia con pasaje de mensajes

Programas distribuidos. Definición.
Relación entre mecanismos de comunicación.
Clases básicas de procesos: productores y consumidores, clientes y servidores, peers.
Control de concurrencia en Sistemas Distribuidos.

Mensajes asincrónicos
Sintaxis y semántica. Canales. Operaciones.
Filtros. Redes de Filtros.
Clientes/Servidores. Algoritmos clásicos. Monitores activos. Continuidad conversacional.
Peers. Intercambio de valores. Cálculo de la topología de una red.
Mensajes asincrónicos en lenguajes reales MPI. Extensión de lenguajes secuenciales con bibliotecas específicas.
Ejemplos.
El concepto de *bag of tasks*. Espacio de tuplas e interacción entre procesos. Estructuras de datos distribuidas. Ejemplos

Mensajes sincrónicos
Sintaxis y semántica.
Conceptos de CSP.
Comunicación guardada. Sintaxis y semántica.
Filtros. Clientes y servidores. Asignación de recursos.
Interacción entre procesos paralelos. Ejemplos.
Mensajes sincrónicos en lenguajes reales.
Ejemplos

Remote Procedure Calls y Rendezvous.
Sintaxis y semántica. Similitudes y diferencias.
RPC: Sincronización en módulos.
Discusión revisada de aplicaciones.
RPC en lenguajes reales: Java. RMI. Ejemplos.
Rendezvous en lenguajes reales: Ada. Tasks y sincronización. Ejemplos

Primitivas múltiples y otros enfoques
Sintaxis y semántica.
MPD. Componentes de programa. Comunicación y sincronización. Ejemplos
Conceptos básicos sobre otros lenguajes/plataformas

Implementaciones
Conceptos de implementación de mecanismos de comunicación y sincronización en ambientes distribuidos.

5. *Paradigmas de interacción entre procesos distribuidos*
Resolución de problemas mediante diferentes paradigmas de interacción entre procesos: Servidores replicados, heartbeat, pipeline, prueba-eco, broadcast, token passing, manager/workers. Ejemplos. Comparación de alternativas.

6. *Introducción a la Programación Paralela*
Objetivos del cómputo paralelo. Necesidad del paralelismo. Areas de aplicación.
Computación científica.
Diseño de algoritmos paralelos.
Métricas. Conceptos de speedup y eficiencia.
La ley de Amdahl y la ley de Gustafson.
Concepto de escalabilidad.

En la práctica se plantean ejercicios que pueden ser realizados como trabajo experimental sobre arquitecturas multiprocesador distribuidas (clusters), multiprocesadores con memoria compartida e híbridos.

BIBLIOGRAFÍA

- Andrews G. “Foundations of Multithreaded, Parallel and Distributed Programming”, Addison Wesley, 2000
- Barnes J., “Programming in Ada 2005 with CD”, Addison Wesley, 2006.
- Ben-Ari, M. “Principles of Concurrent and Distributed Programming, 2/E”. Addison-Wesley. 2006. ISBN 0-321-31283-X
- Chandy, Misra, “Parallel Program Design. A Foundation”, Addison Wesley, 1988.
- Downey, Allen. “The Little Book of Semaphores, Second Edition”. Free book disponible en http://www.freetechbooks.com/the-little-book-of-semaphores-second-edition-t519.html, 2007.
- Filminas de las clases teóricas.
- Grama A., Gupta A., Karypis G., Kumar V., "An Introduction to Parallel Computing. Design and Analysis of Algorithms", Pearson Addison Wesley, 2nd Edition, 2003
- Herlihy M., Shavit N., “The Art of Multiprocessor Programming”. Morgan Kaufmann. Revised reprint, 2012.
- Hoare C., “Communicating Sequential Processes”, Englewood Cliffs, Prentice Hall, 1985
- Brinch Hansen, P., “Studies in Computational Science. Parallel Programming Paradigms”, Prentice Hall, 1995.
- Jordan H.F., Alaghband G., Jordan H.E., "Fundamentals of Parallel Computing", Prentice Hall, 2002
- Naiouf, De Giusti A., De Giusti L, Chichizola, “Conceptos de concurrencia y paralelismo”, UNLP, a publicar 2017.
- Pacheco, P. “An introduction to parallel programming”. Morgan Kaufmann, 2011.
- Raynal M. “Concurrent Programming: Algorithms, Principles, and Foundations”. Springer, 2012.
- Taubenfeld, Gadi. “Synchronization Algorithms and Concurrent Programming”. Prentice Hall. 2006.

METODOLOGÍA DE ENSEÑANZA

La actividad curricular se organiza en:

- clases teóricas, a cargo de un profesor, en las cuales se imparten los temas de la asignatura. Los alumnos podrán asistir a cualquiera de los dos horarios.
- explicaciones de práctica, a cargo de un profesor y/o JTP, y que actúan a modo de articulación entre teoría y práctica y donde se plantean y resuelven problemas “tipo”.
- clases prácticas, a cargo de auxiliares docentes (ayudantes coordinados por JTP), donde los alumnos trabajan sobre los ejercicios propuestos en la guía de trabajos prácticos. Los alumnos pueden asistir indistintamente a cualquiera de los horarios de práctica.
- clases de consulta (de teoría y práctica).
- periódicamente se publican cuestionarios de teoría a modo de guía a fin de que los alumnos reflexionen sobre los puntos más importantes.

El reglamento y cronograma tentativo son conocidos por los alumnos desde el inicio de la actividad curricular.

Se utiliza un entorno virtual de enseñanza-aprendizaje (IDEAS), donde se encuentran disponibles clases, guías de TP, avisos, resultados de exámenes, etc.

Para las clases teóricas y las explicaciones de práctica se utiliza PC, cañón y pizarrón.

Los alumnos pueden realizar prácticas en PC usando distintos lenguajes/bibliotecas que soportan concurrencia.

EVALUACIÓN

La cursada puede obtenerse de dos maneras:
a) Aprobar un parcial normal en alguna de las 3 fechas que se dispondrán (parcial y 2 recuperatorios)
b) Alternativamente y de tipo opcional, se cuenta con un régimen de promoción con las siguientes características:
- se tomarán durante el curso 2 parcialitos prácticos (uno correspondiente a memoria compartida y uno a memoria distribuida)
- si el alumno aprueba ambos parcialitos obtiene la cursada
- si el alumno aprueba uno de los parcialitos podrá obtener la cursada aprobando un parcial reducido que se rinde en las mismas fechas que el normal.
Se considerarán en condición de "Desaprobado" los alumnos que no obtengan la cursada y que rindan y obtengan "D" al menos en 2 de las 3 fechas de evaluación parcial.

El final puede aprobarse de dos maneras:
a) Final tradicional (teórico-práctico) en mesa de finales.
b) Alternativamente y de tipo opcional, cumpliendo las siguientes condiciones (cada una tiene como precondición cumplir con la anterior):

i) Rendir los 2 parcialitos teóricos que se tomarán en las clases
ii) Luego de aprobar la cursada, rendir y aprobar un parcial teórico.
iii) Cumplido ii):
- si la nota del parcial teórico es >= 7 el alumno queda habilitado para rendir un coloquio en mesa de final dentro del semestre siguiente.
- si la nota del parcial teórico es >=4 y <7, se asigna un trabajo individual. El mismo debe ser desarrollado y defendido en un coloquio en fecha de final dentro del semestre siguiente.

En caso de presentarse a rendir final tradicional, la promoción de teoría caduca.

Si bien se toman en los mismos días, los parcialitos de práctica y de teoría son independientes (el alumno puede rendir uno de los dos o ambos).

CRONOGRAMA DE CLASES Y EVALUACIONES

El comienzo de clases está previsto para la semana del 5 de marzo.
El cronograma detallado se pone en conocimiento de los alumnos al inicio del curso.
El esquema preliminar por bloque de las clases y evaluaciones es el siguiente:

Clase	Contenidos/Actividades
1 y 2	Conceptos básicos. Comunicación y sincronización. Interferencia.
3 a 6	Concurrencia con memoria compartida.
7 a 9	Concurrencia con memoria distribuida
10	Introducción a la Programación Paralela.

Evaluaciones previstas	Fecha estimada
1er parcialito	Semana del 23 o 30/4
2do parcialito	Semana del 21 o 28/5
Parcial	Semana del 11/6
1er recuperatorio	Semana del 25/6
2do recuperatorio	Semana del 9/7
Parcial teórico	Semana del 6/8

Contacto de la cátedra (mail, página, plataforma virtual de gestión de cursos):

Plataforma virtual: ideas.info.unlp.edu.ar
Web: weblidi.info.unlp.edu.ar/catedras/concurrente/WEB/
Mail: mnaiouf@lidi.info.unlp.edu.ar, ldgiusti@lidi.info.unlp.edu.ar, francoch@lidi.info.unlp.edu.ar

Firma del/los profesor/es