PROGRAMA DE CURSO
SISTEMAS OPERATIVOS

A. Antecedentes generales del curso:

Departamento	Ciencias de la Computación							
Nombre del curso	Sistemas Operativos		Código		CC4302	Créditos		6
Nombre del curso en inglés	*Operating Systems*							
Horas semanales	Docencia	3	Auxiliares	1,5		Trabajo personal	5,5	
Carácter del curso	Obligatorio	X		Electivo				
Requisitos	CC3301: Programación de Software de Sistemas							

B. Propósito del curso:

El propósito del curso sistemas operativos es que los/las estudiantes escriban programas que hagan un uso eficiente de los procesadores multi-core y que sean capaces de evaluar el impacto que tienen en el desempeño del software, la arquitectura del hardware y las estrategias que usa el núcleo del sistema operativo para administrar los recursos de hardware (CPU, memoria primaria y disco/ssd).

El curso tributa a las siguientes competencias específicas (CE) y genéricas (CG):

CE2: Analizar, diseñar y/o adoptar, algoritmos y estructuras de datos que cumplan con las garantías requeridas de correctitud y eficiencia.

CE6: Desarrollar software en una amplia variedad de plataformas y lenguajes de programación.

CE8: Diagnosticar y resolver problemas en el funcionamiento de software cercano a la plataforma para mejorar su desempeño.

CG2: Comunicación en inglés

Leer y escuchar de manera comprensiva en inglés una variedad de textos e informaciones sobre temas concretos o abstractos, comunicando experiencias y opiniones, adecuándose a diferentes contextos y a las características de la audiencia.

C. Resultados de aprendizaje:

Competencias específicas	Resultados de aprendizaje
CE2, CE6	RA1: Desarrolla programas concurrentes (con múltiples threads), utilizando herramientas de sincronización como semáforos, mutex y condiciones, a fin de hacer un uso eficiente de los procesadores multi-core.
CE8	RA2: Determina cómo es el funcionamiento del núcleo del sistema operativo, como ejemplo eficiente de un sistema concurrente que favorece la resolución de problemas similares que se presentan en el desarrollo de aplicaciones concurrentes.
CE8	RA3: Evalúa el impacto en el desempeño del software que tiene la manera en que el núcleo del sistema operativo administra la memoria primaria y el disco o ssd. RA4: Concibe e implementa soluciones eficientes en el uso de memoria primaria y secundaria, considerando el paginamiento en demanda y la estructura del sistema de archivos.
CE2	RA5: Paraleliza y sincroniza con múltiples threads cuando no se cumple con la exigencia de tiempos de ejecución con un programa secuencial, escribiendo programas que hagan un uso eficiente de la memoria virtual provista por el sistema de paginamiento.
Competencias genéricas	**Resultados de aprendizaje**
CG2	RA6: Lee textos en inglés, integrando a su formación el uso de términos técnicos, aplicables al ámbito de sistemas operativos.

D. Unidades temáticas:

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
1	RA1, RA5, RA6	Procesos, threads y sincronización	4,5 semanas
Contenidos		**Indicador de logro**	
1.1. Historia de los sistemas operativos: 1.1.1. Sistemas batch, job, dump, monitor residente, operación off-line. 1.1.2. Multi-programación, job scheduling, máquinas virtuales, procesos. 1.1.3. Computadores personales, redes, sistemas distribuidos. 1.2. Procesos pesados vs. procesos livianos o threads. 1.3. Preemption vs. non-preemption. 1.4. Un sistema de procesos livianos: pthreads. 1.5. Paralelización por medio de threads. 1.6. Errores de la programación con threads: dataraces, deadlocks y starvation (hambruna). 1.7. Secciones críticas y exclusión mutua. 1.8. Sincronización de threads mediante semáforos, mutex y condiciones. 1.9. Problemas clásicos de sincronización de procesos: productor/consumidor, cena de filósofos, lectores/escritores.		La/el estudiante: 1. Explica cómo surgieron los primeros sistemas operativos y de dónde salen palabras y conceptos relacionados con los sistemas operativos. 2. Desarrolla programas con múltiples threads paralelos, considerando cuando sus contrapartes secuenciales no alcanzan los requerimientos de desempeño impuestos por el usuario. 3. Resuelve problemas que requieren sincronizar los threads. 4. Usa semáforos, mutex y condiciones para evitar errores de programación como *dataraces*, *deadlocks* y *starvation*. 5. Usa el *debugger* y herramientas como *valgrind/drd* para detectar errores de manejo de threads. 6. Lee artículos en inglés, considerando la adquisición y uso de términos técnicos aplicables a threads.	
Bibliografía de la unidad		[4] Capítulo 1.2. [2] y [3] Capítulos 5.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
2	RA2, RA6	Administración del Procesador	4 semanas

Contenidos	Indicador de logro
2.1. Estructura del sistema operativo: núcleo, llamadas de sistema, API (application programming interface). 2.2. Conceptos básicos para la implementación del núcleo de un sistema operativos: identificador de proceso, descriptor de proceso, estados de un proceso, cambio de contexto, ráfagas de CPU, tiempo de despacho y tiempo de respuesta, núcleos físicos de ejecución (cores). 2.3. Estrategias de scheduling del procesador: first come first served, shortest job first, colas de prioridad, round robin, scheduling en varios niveles. 2.4. Requerimientos de hardware: cronómetro regresivo e interrupciones. 2.5. Núcleo clásico vs. núcleo moderno. 2.6. Sincronización básica de cores por medio de spin-locks. 2.7. Ejemplo de implementación de: un scheduler simple, semáforos, mutex y condiciones.	La/el estudiante: 1. Evalúa el impacto que tienen las distintas estrategias de scheduling de procesos a nivel del núcleo del sistema operativo, en el tiempo de respuesta, tiempo de despacho y sobrecosto del cambio de contexto al ejecutar los programas de los usuarios. 2. Evalúa el hardware de una CPU, determinando qué características se pueden implementar a nivel de su sistema operativo, como protección entre usuarios, atención concurrente a los eventos gatillados por distintos dispositivos, etc. 3. Compara un núcleo moderno versus un núcleo clásico, evaluando ventajas y desventajas del núcleo moderno, en base a aspectos como el número de cores de la CPU y la complejidad de su código fuente. 4. Implementa herramientas de sincronización de threads similares a los semáforos a partir de problemas que se le presentan. 5. Lee en inglés textos, integrando nuevos términos técnicos, aplicables al ámbito de sistemas operativos.
Bibliografía de la unidad	[3] Capítulo 6.

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
3	RA3, RA4, RA5	Administración de memoria	3 semanas

Contenidos	Indicador de logro
3.1. Implementación de espacios de direcciones virtuales por medio de paginamiento. 3.2. Requerimientos de hardware: modo dual, la MMU (memory management unit), tablas de páginas, atributos de una página, traducción de direcciones virtuales. 3.3. El potencial del paginamiento: protección entre procesos, extensión del área de datos y la pila, implementación eficiente de fork, mmap y swapping. 3.4. Paginamiento en demanda: estrategias de reemplazo de páginas, la estrategia del reloj y la estrategia del working set. 3.5. Paginamiento en la x86 y x86-64, tablas de páginas de múltiples niveles. 3.6. Comparación de paginamiento con segmentación. 3.7. Implementación de máquinas virtuales.	La/el estudiante: 1. Escribe programas eficientes en el uso de memoria y tiempo de ejecución, considerando la manera en que el núcleo del sistema operativo administra la memoria. 2. Evalúa el desempeño de las 2 principales estrategias de paginamiento en demanda, reloj y working set, considerando sobrecosto en tiempo de ejecución en casos de uso extremo de memoria y en casos de bajo uso de memoria. 3. Decide, en base a una evaluación, cuándo es conveniente recurrir a un sistema operativo virtualizado o cuando es mejor instalarlo de manera nativa, fundamentando su decisión. 4. Evalúa los sobrecostos de las tablas de páginas para administrar la memoria en términos de tiempo de ejecución y de uso de memoria adicional.
Bibliografía de la unidad	[3] Capítulo 6.

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
4	RA3, RA4	Entrada/Salida	3,5 semanas

Contenidos	Indicador de logro
4.1. Módulos y drivers de Linux. 4.2. Requerimientos de hardware: entrada/salida mapeada en la memoria, interrupciones, canales. 4.3. Arquitectura en capas del sistema de entrada: sistema de archivos, caché de disco, scheduling de disco, driver y disco. 4.4. Discos vs. SSD (solid state drive). 4.5. El sistema de archivos de Unix. 4.6. Estrategias de scheduling de disco: first come first served, shortest seek first y el método del ascensor.	La/el estudiante: 1. Programa módulos y drivers de Linux para, por ejemplo, acceder a un nuevo dispositivo de entrada/salida (como una impresora, joystick, etc.). 2. Escribe programas que realizan un acceso eficiente a los archivos, considerando su organización en el disco o ssd. 3. Decide cuándo se debe recurrir a un ssd y cuándo a un disco, considerando las ventajas y desventajas de cada uno. 4. Determina y explica los beneficios del diseño en capas, en cuanto a simplicidad, modularidad, mantenibilidad, etc., comparado con un diseño monolítico (todo mezclado).
Bibliografía de la unidad	[3] Capítulos 10 al 13

E. Estrategias de enseñanza - aprendizaje:

El curso considera diversas estrategias de enseñanza:

- Clases expositivas.
- Resolución de problemas simples.

F. Estrategias de evaluación:

Para esta propuesta se recomiendan las siguientes instancias de evaluación. De todas formas, es necesario señalar que al inicio del semestre se informará sobre el tipo de evaluación y la ponderación que se asignará a cada evaluación.

Tipo de evaluación	Unidades asociadas a la evaluación
Controles de materia	Control 1: evalúa unidad 1. Control 2: evalúa unidad 2. Control 3: evalúa unidades 3 y 4.
Examen final	Todas las unidades.
Aproximadamente 6 tareas	Tarea 1: Paralelización con threads. Tarea 2: Sincronización de threads con mutex y una condición. Tarea 3: Sincronización de threads con múltiples condiciones y semáforos. Tarea 4: Implementación de herramientas de sincronización. Tarea 5: Implementación de *timeouts.* Tarea 6: Implementación de un driver en Linux.

G. Recursos bibliográficos:

Bibliografía obligatoria:

[1] Mateu, L. “Apuntes de sistemas operativos”, https://users.dcc.uchile.cl/~lmateu/CC4302

Bibliografía complementaria:

[2] B. Nichols, D. Buttlar, J. Proulx (1996), “Pthreads Programming: A POSIX Standard for Better Multiprocessing”, O´Reilly, ISBN: 1-56592-115-1
[3] A Silberschatz et al (2012,2013), 9ª edición, “Operating System Concepts”, Wiley, ISBN: 978-1118063330.
[4] A Tanenbaum, Herbert Bos (2015), 4ta edición, “Modern Operating Systems”, Prentice Hall, ISBN: 978-0133591620.

H. Datos generales sobre elaboración y vigencia del programa de curso:

Vigencia desde:	Primavera, 2021
Elaborado por:	Luis Mateu
Validado por:	Revisión y validación entre académicos: Javier Bustos, Sergio Ochoa, José Piquer y validación CTD de Computación
Revisado por:	Área de Gestión Curricular