PROGRAMA DE CURSO
PROGRAMACIÓN DE SOFTWARE DE SISTEMAS

A. Antecedentes generales del curso:

Departamento	Ciencias de la Computación					
Nombre del curso	Programación de Software de Sistemas		Código	CC3301	Créditos	6
Nombre del curso en inglés	*Systems Programming*					
Horas semanales	Docencia	3	Auxiliares	1,5	Trabajo personal	5,5
Carácter del curso	Obligatorio	X		Electivo		
Requisitos	CC3001: Algoritmos y Estructuras de Datos					

B. Propósito del curso:

El propósito del curso Programación de software de sistemas es que los/las estudiantes escriban y mejoren programas en lenguaje C, que requieren hacer un uso eficiente de la plataforma y, por lo tanto, necesitan conocer su arquitectura de hardware y la interfaz de programación de aplicaciones (API) del sistema operativo.

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
CE6	RA1: Desarrolla programas en lenguaje C, eficientes en tiempo de ejecución y uso de memoria, haciendo uso de la interfaz de programación de aplicaciones (API) de Linux.
CE8, CE2	RA2: Evalúa el impacto de las características del hardware y de los programas en el desempeño del computador, considerando el tiempo de ejecución y/o el consumo energético.

CE8	RA3: Usa la herramienta de depuración de programas (debugger) para diagnosticar los errores de programación a nivel de lenguaje C y a nivel de lenguaje assembler.
CE2	RA4: Escribe programas correctos que cumplan exigencias de tiempos de ejecución y de uso de memoria para un mejor desempeño.
Competencias genéricas	Resultados de aprendizaje
CG2	RA4: Lee textos en inglés, integrando a su formación el uso de términos técnicos, aplicables a la programación de software de sistemas.

D. Unidades temáticas:

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
1	RA1, RA3, RA4	Programación eficiente en lenguaje C	7 semanas
Contenidos		Indicador de logro	
1.1. Principios básicos del lenguaje C. 1.2. Sistema de tipos. 1.3. Operaciones con *bits*. 1.4. Variables, arreglos, punteros, *strings*. 1.5. Definición de nuevos tipos con *struct* y/o *typedef*. 1.6. Punteros a funciones. 1.7. Archivos binarios o de texto, de acceso secuencial o directo. 1.8. Excepciones: *Setjmp/longjmp.* 1.9. Etapas de la compilación.		La/el estudiante: 1. Desarrolla programas correctos y eficientes en tiempos de ejecución y uso de memoria en lenguaje C. 2. Usa el *debugger* para diagnosticar los errores de programación a nivel de lenguaje C. 3. Programa usando estructuras de datos eficientes en tiempo de ejecución. 4. Compara los tiempos de ejecución y el uso de memoria de los programas desarrollados. 5. Programa correctamente evitando errores tales como inconsistencia de tipos, *referencias colgantes y fugas de memoria*. 6. Maneja archivos binarios o de texto, de acceso secuencial o directo. 7. Programa código genérico por medio de punteros a funciones. 8. Identifica en qué nivel se producen los errores: pre-proceso, compilación, link. 9. Lee artículos en inglés, considerando el uso de términos técnicos sobre lenguaje C.	
Bibliografía de la unidad		[1] Capítulo 2.	

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
2	RA2, RA3, RA4	Arquitectura de computadores	4 semanas
Contenidos		Indicador de logro	
2.1. Arquitecturas lógicas.		La/el estudiante:	

2.1.1. Computadores con conjunto de instrucciones complejas (CISC) vs. computadores con conjunto de instrucciones reducido (RISC). 2.1.2. *Assembler*. 2.2. Sistemas digitales. 2.2.1. Compuertas básicas: and, or, not, reloj 2.2.2. Líneas de datos, buses. 2.2.3. Componentes modulares: decodificadores, multiplexores, registros, memoria, sumadores, ALU, unidad de control. 2.3. Arquitectura física del procesador: 2.3.1. Diseño básico. 2.3.2. Etapas de la ejecución de una instrucción: *fetch*, *decode*, *execute*, etc. 2.3.3. Memoria cache. 2.3.4. Arquitecturas microprogramadas, en *pipeline*, *superescalares*, con ejecución fuera de orden.	1. Escribe programas sencillos en *assembler*. 2. Diagnostica y depura los errores de programación en assembler. 3. Identifica qué instrucciones en assembler se corresponden con la instrucción respectiva del código fuente en lenguaje C. 4. Compara las ventajas y desventajas de RISC por sobre CISC. 5. Evalúa cada arquitectura del procesador en términos de eficiencia, tiempo de ejecución y eficiencia energética. 6. Lee artículos en inglés, considerando la adquisición de términos técnicos sobre arquitectura de computadores.
Bibliografía de la unidad	[2] Capítulos 2 y 5.

Número	RA al que tributa	Nombre de la unidad	Duración en semanas
3	RA1, RA2, RA4	Sistema operativo Linux	4 semanas

Contenidos	Indicador de logro
3.1. Historia de Unix y Linux. 3.2. Uso de la interfaz de programación de aplicaciones (*API)* de Linux para manejo de archivos y directorios. 3.3. Procesos pesados: *fork/exec/pipes.* 3.4. El intérprete de comandos: shell. 3.5. Señales.	La/el estudiante: *1.* Escribe programas que acceden a archivos y directorios por medio de la *API* de Linux. 2. Escribe un pequeño intérprete de comandos por medio de *fork* y *exec*. 3. Usa *pipes* para conectar la salida de un programa con la entrada de otro programa. 4. Paraleliza programas por medio de múltiples procesos pesados. 5. Compara y evalúa tiempos de ejecución de programas secuenciales y programas paralelizados, tanto en multiprocesadores como en monoprocesadores. 6. Captura señales que indican eventos como cronómetros regresivos que llegan a cero, interrupciones del usuario, violación de segmentos, división por cero, etc. 7. Lee artículos en inglés considerando el uso de términos técnicos de los sistemas operativos Unix, Linux y su historia.
Bibliografía de la unidad	[1] Capítulo 4

E. Estrategias de enseñanza - aprendizaje:

El curso considera diversas estrategias de enseñanza:

- Clases expositivas.
- Programación de problemas simples.

F. Estrategias de evaluación:

Para esta propuesta se recomiendan las siguientes instancias de evaluación. De todas formas, es necesario señalar, de todas formas 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 unidad 3.
Examen final	Todas las unidades.
Aproximadamente 9 mini-tareas	Mini-tareas 1 a 5: lenguaje C. Mini-tareas 6 y 7: arquitectura de computadores. Mini-tareas 8 y 9: sistema operativo Linux.

G. Recursos bibliográficos:

Bibliografía obligatoria:

[1] Mateu, L. "Apuntes de programación de software de sistemas", https://wiki.dcc.uchile.cl/cc3301/temario.

[2] Guerrero, P. ``Apuntes de arquitectura de computadores", https://users.dcc.uchile.cl/~lmateu/CC4301/download/Apuntes-Pablo-Guerrero.pdf.

Bibliografía complementaria:

[3] Kernighan, B. y Ritchie, D. (1988). ``The C Programming Language´´, Prentice-Hall, 2nd edition.

[4] Stones, R., Matthew, N. (2007). "Beginning Linux Programming (Programmer to Programmer)". Wiley, 4th edition.

[5] Patterson, D.A. and John L. Hennessy, (2008), "Computer Organization and Design: The Hardware/Software Interface", Elsevier, 4th edition.

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, Patricio Poblete, Sergio Ochoa, José Piquer y validado por CTD de Computación
Revisado por:	Área de Gestión Curricular