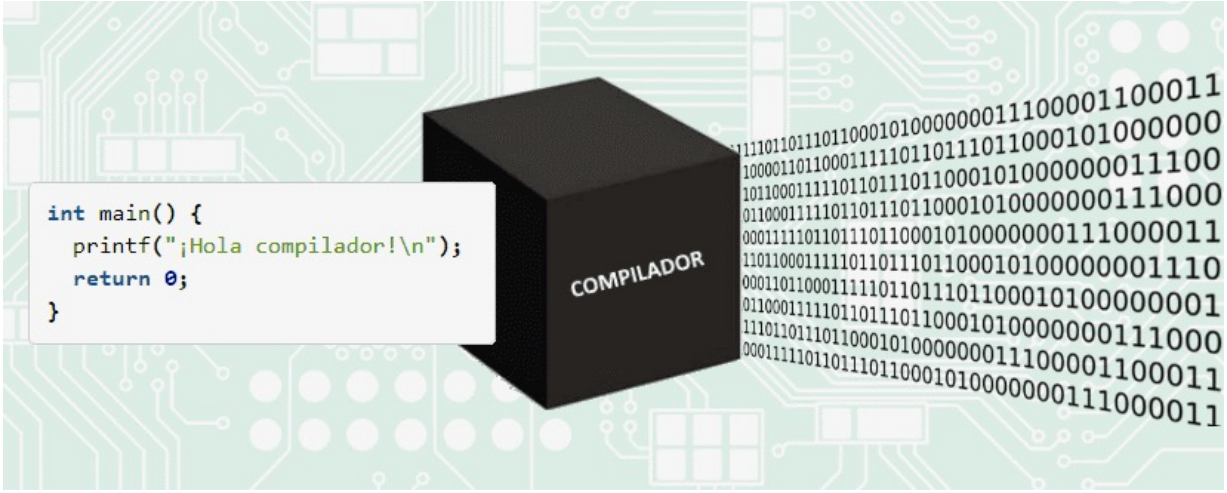
Programación de sistemas de base I

Dr. José Lázaro Martínez Rodríguez

Introducción

- Un ingeniero en sistemas aborda diversos aspectos computacionales para el desarrollo de sistemas y aplicaciones computacionales
- Uno de estos aspectos es el desarrollo de compiladores para poder preparar los programas para ser ejecutados por una computadora bajo cierta arquitectura



Introducción

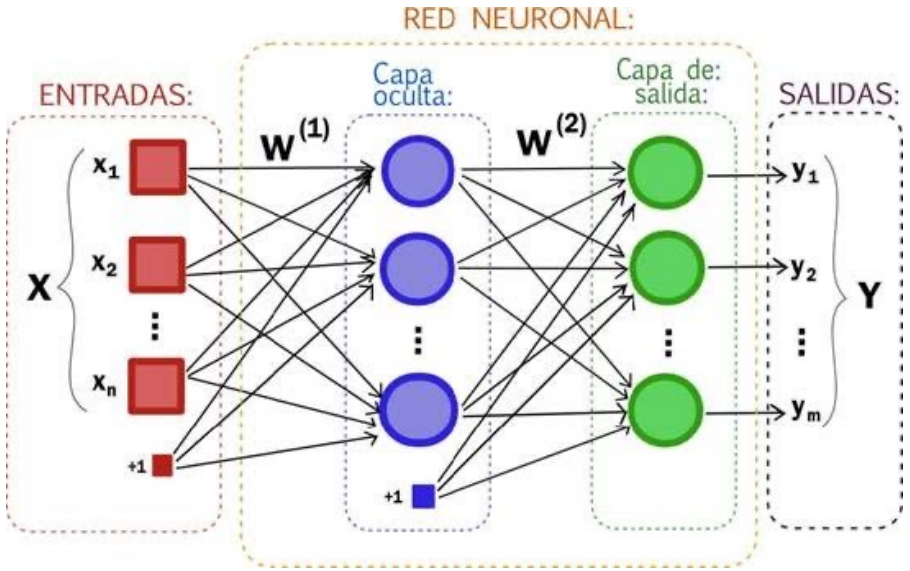
- Los principios y técnicas que se usan en la escritura de compiladores se pueden emplear en muchas otras áreas.
- Se basan en los conceptos de teoría de autómatas y lenguajes formales que se están exponiendo en la parte teórica, y consituyen un campo de aplicación práctica bastante directo.

Introducción

- Los principios y técnicas que se usan en la escritura de compiladores se pueden emplear en muchas otras áreas.
- Se basan en los conceptos de teoría de autómatas y lenguajes formales que se están exponiendo en la parte teórica, y consituyen un campo de aplicación práctica bastante directo.

Introducción

- Los principios y técnicas que se usan en la escritura de compiladores se pueden emplear en muchas otras áreas.
- Se basan en los conceptos de teoría de autómatas y lenguajes formales que se están exponiendo en la parte teórica, y consituyen un campo de aplicación práctica bastante directo.



¿Por qué un compilador?

- Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar

¿Por qué estudiar lenguajes y compiladores?

- Aumentar la capacidad de expresión
- Mejorar la comprensión del comportamiento del programa
- Aumentar la capacidad de aprender nuevos lenguajes

- Aprende a construir un sistema grande y fiable
- Ver muchos conceptos básicos de CS en funcionamiento

Objetivo

- Que el estudiante comprenda las etapas involucradas en un compilador y su asociación con otras tareas en el desarrollo de sistemas computacionales

Panorama

1. Introducción a la compilación
2. Fundamentos de compiladores
3. Análisis léxico
4. Análisis sintáctico
5. Análisis semántico

Introducción a la compilación

- Se ofrece una descripción inicial de todos los elementos por estudiar, sus fases y la relación que tienen los temas/parciales de la materia.
- La escritura de compiladores comprende los lenguajes de programación, la arquitectura de computadoras, la teoría de lenguajes, los algoritmos y la ingeniería de software.

Fundamentos de compiladores

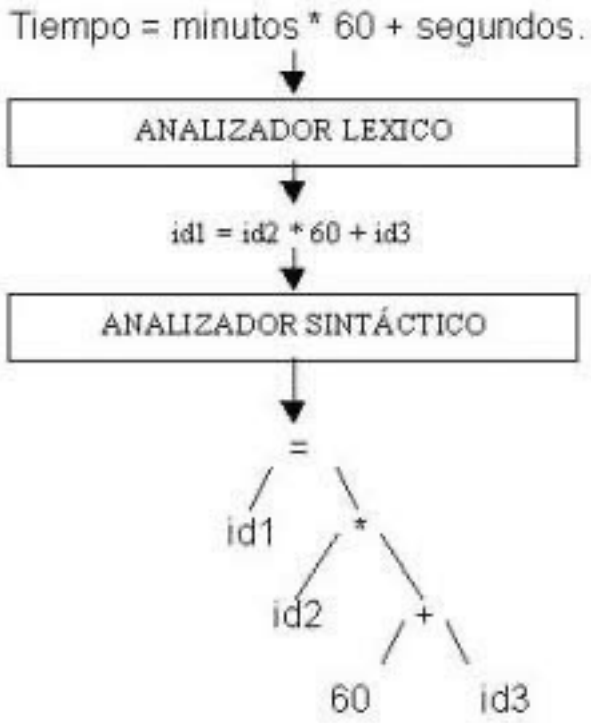
- Conocerá los elementos de la programación de sistemas, tales como cargadores, ensambladores, macro procesadores y sistemas operativos.

Análisis léxico

- La cadena de entrada se recibe como una sucesión de caracteres.
- El análisis léxico agrupa los caracteres en secuencias con significado colectivo y mínimo en el lenguaje, llamadas componentes léxicos (palabras o “token”), con ciertos atributos léxicos
- Se estudiarán además lo que son las expresiones regulares

Análisis sintáctico

- Normalmente las frases se representan mediante una estructura de árbol sintáctico, siguiendo reglas que describen el lenguaje.



Evaluación

Se requieren conocimientos básicos de programación

- Primeros dos parciales (24 feb, 28 abril -tentativamente-)
  - Examen 50 %
  - Prácticas/tareas 30 %
  - Asistencia y participación 20% (se acreditará con más del 90% de asistencia)
  - Reprobar dos parciales causa reprobación de la materia
- Adicionalmente se considera un proyecto final que contará como el 100% del tercer parcial (no entregarlo provoca reprobación de la materia)
- Extraordinario (23-27 may)

Reglas

- Asistir puntualmente
- Seguir Normas de salud
- *No introducir alimentos a la sala de cómputo*
- *No introducir mochilas (sala cómputo)*
- Evitar el uso de dispositivos móviles para propósitos diferentes a la clase
- Ser respetuoso con sus compañeros de clase
- Poner atención para contestar preguntas
- Prohibido el plagio

Reglas

- Se requiere al menos enviar el 80 % de tareas para tener derecho a examen de recuperación
- Cumplir con asistencia para tener derecho a cualquier examen
- El examen regularmente se debe hacer en el cuaderno, a mano, y enviar capturas de sus respuestas en formato PDF (por Teams)
- Indispensable contar con equipo de cómputo para realizar prácticas de programación

Bibliografía

- 1. Alfred V . Aho Compiladores principio, técnicas y Herramientas. Ed. Addison – Wesley Iberoamericana.
- 2. García Pedro, Teoría de Autómatas y Lenguajes formales. Ed Alfaomega
- 3. John E. Hopcropft, Jeffrey D. Ullman Introduccion a la teoría de lenguajes y Computación Ed. Addison – Wesley Iberoamericana.
- 4.- Van Gigch, John P. Teoría General de Sistemas. Ed. Trillas.
- 5. Alfonseca Moreno, Manuel Compiladores e Intérpretes: Teoría y Práctica. Ed. Pearson
- 6. Kelley,Dean Teorí a de Autómatas y Lenguajes Formales , Prentice Hall
- 7. Terrence W. Pratt Lenguajes de Programación. Prentice – Hall
- 8. Brian W. Kerninghan Lenguajes de Programación. Prentice – Hall
- 9- Jacobson, Ivarei Proceso Unificado de Desarrollo de SoftwareAddison Wesley
- 10. Pressman, Roger S , Ingenieria de Software. Un Enfoque Práctico. Mc Graw Hill.(2002) 11. Sommerville, lan (2002) Ingenieria de Software 6a .ed Pearson Education , Néxico, 712 p.