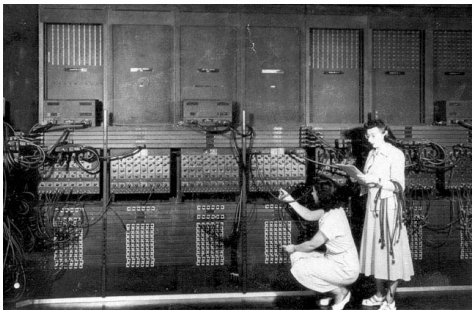
Procesadores de lenguaje
Tema 1 – Introducción a los compiladores

Salvador Sánchez, Daniel Rodríguez
Departamento de Ciencias de la Computación
Universidad de Alcalá

Resumen del tema

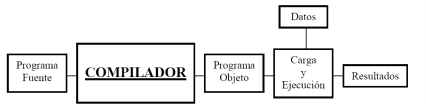
- Traductores
- Estructura de un compilador
- Descripción general de las fases de compilación
  - Fase de análisis (front-end)
  - Fase de síntesis (back-end)
  - Agrupamiento de las fases
- Aplicaciones prácticas

Evolución de los compiladores



Traductores

- Los lenguajes permiten la comunicación.
- El receptor de un mensaje debe entenderlo y actuar en consecuencia, para ello debe antes procesarlo.
- Transforma un texto escrito en un lenguaje (fuente) a otro texto en un lenguaje distinto (objeto), manteniendo el significado del texto inicial.



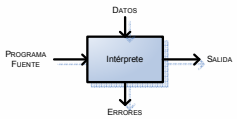
La necesidad de traducción

Clasificación lenguajes de programación

- Lenguajes máquina
  - Son los lenguajes de más bajo nivel: secuencias binarias de ceros y unos.
  - Históricamente, los primeros
- Lenguajes ensambladores
  - Segunda generación de lenguajes
  - Versión simbólica de los lenguajes máquina (MOV, ADD, etc).
- Lenguajes de alto nivel
  - Lenguajes de tercera generación (3GL)
    - Estructuras de control, Variables de tipo, Recursividad, etc.
    - Ej.: C, Pascal, C++, Java, etc
- Lenguajes orientados a problemas.
  - Lenguajes de cuarta generación (4GL)
    - Ej. SQL

Tipos de traductor: intérprete

- El **intérprete** ejecuta el código según lo va interpretando.
  - Cada vez que se escribe una línea el programa comprueba si es correcta, si lo es, la ejecuta.
  - La ejecución es interactiva.
  - Los intérpretes más puros no guardan copia del programa que se está escribiendo.
  - Ejemplos: Procesos por lotes, CAML, etc.
  - El intérprete siempre debe estar presente.



Tipos de traductor: compilador

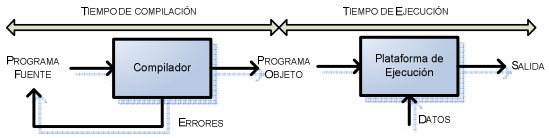
- El **compilador** es el traductor más extendido
- Realiza un análisis y genera un programa ejecutable
- El programa ejecutable, una vez creado, no necesita el compilador para funcionar

Compilador vs. intérprete

- Compilador
  - Se compila una vez, se ejecuta *n* veces
  - El proceso de compilación tiene una visión global de todo el programa, por lo cual la gestión de errores es más eficiente.
  - La ejecución es más rápida.
- Intérprete
  - Se traduce cada vez que se ejecuta
  - Permite interaccionar más con el código en tiempo de ejecución.
  - Necesita menos memoria.

Fases de un programa

- La vida de un programa, desde que se escribe hasta que se ejecuta en una plataforma, se pueden distinguir dos periodos de tiempo:
  - Tiempo de compilación: Periodo en el que el programa fuente se traduce al programa objeto equivalente.
  - El tiempo de ejecución: Tiempor durante el cual el programa objeto se ejecuta sobre una plataforma específica.

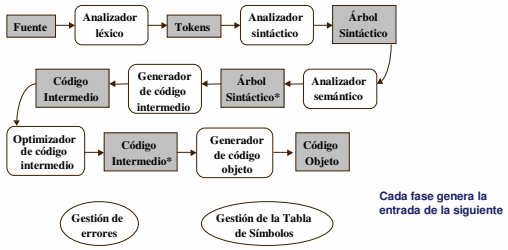


Tareas de un compilador

- Comprobación de los **elementos del lenguaje**: ¿Qué elementos están presentes?
- Comprobación de la **combinación** de los elementos del lenguaje: ¿Es correcta su combinación?
- Comprobación del **significado** de los elementos del lenguaje: ¿El significado de lo especificado es válido?
- Generación de instrucciones para una máquina.

AQUI

Estructura de un compilador (detalle)



Cada fase genera la entrada de la siguiente

Otros elementos

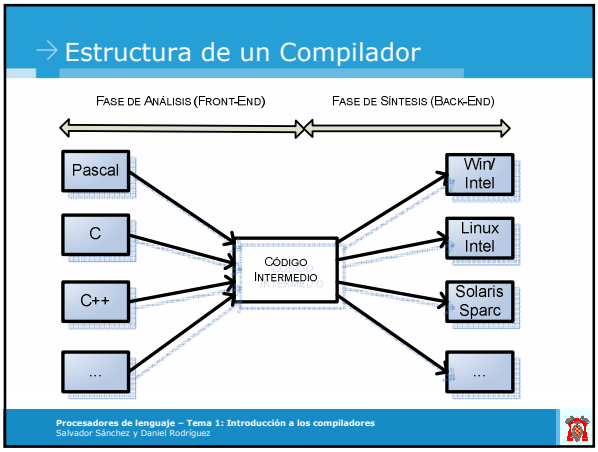
- **Tabla de símbolos**: estructura de datos que contiene todos los identificadores reconocidos por el compilador.
  - El gestor de la tabla de símbolos interactúa con cada una de las fases del compilador, consultando, añadiendo y modificando su información.
- El **Precompilador** es un módulo opcional que se ejecuta antes de invocar al compilador:
  - El programa fuente incluye estructuras, instrucciones o declaraciones escritas en otro lenguaje, denominado lenguaje empotrado.
  - Todas las instrucciones del lenguaje empotrado deben ser traducidas a instrucciones del lenguaje anfitrión para que puedan ser compiladas.

Gestión de errores

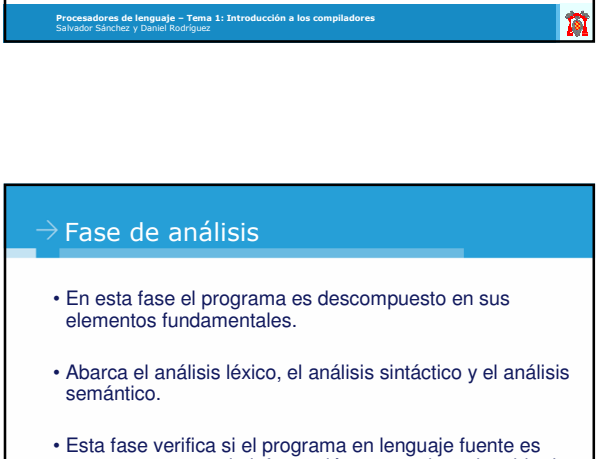
- Se utiliza en todas las fases, pero especialmente en las de análisis sintáctico y semántico.
- Es una tarea difícil:
  - Un error puede ocultar otros
  - Un error puede provocar una avalancha de otros
- Importante tener un buen manejo de errores:
  - Que detecte todos los errores
  - Que no deje de compilar al detectar el primer error

Estructura de un Compilador

- El desarrollo de un compilador es complejo, generalmente, se simplifica si se divide en dos fases enlazadas por el código intermedio.
- Problema: *m* lenguajes fuente y *n* lenguajes objeto, *m* x *n* compiladores diferentes o bloques
- Solución: dividir el compilador en 2 fases
  - **Fase de Análisis (front-end)**: depende del lenguaje fuente, independiente del lenguaje objeto.
  - **Fase de Síntesis (Back-end)**: depende del lenguaje objeto y es independiente del lenguaje fuente.
  - De esta manera, tenemos m fases de análisis y n fases de síntesis -> *m+n* bloques

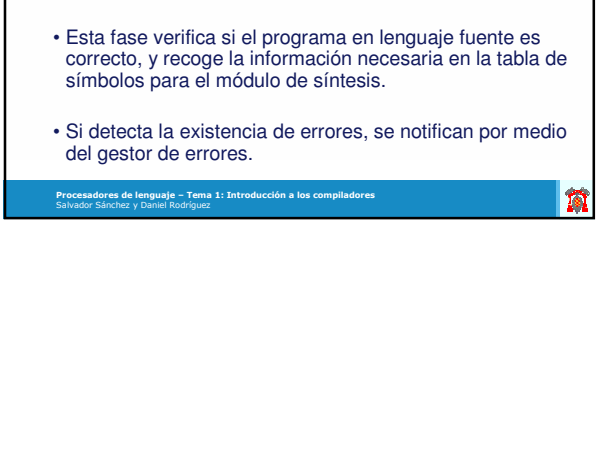


Estructura de un Compilador



Fase de análisis

- En esta fase el programa es descompuesto en sus elementos fundamentales.
- Abarca el análisis léxico, el análisis sintáctico y el análisis semántico.
- Esta fase verifica si el programa en lenguaje fuente es correcto, y recoge la información necesaria en la tabla de símbolos para el módulo de síntesis.
- Si detecta la existencia de errores, se notifican por medio del gestor de errores.



Fase de síntesis

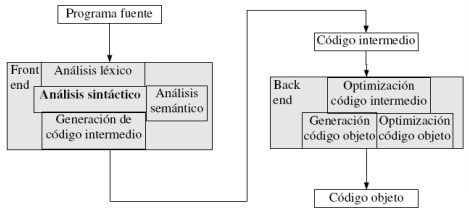
- Esta fase se lleva a cabo una vez que el módulo de análisis ha verificado que el código fuente es correcto.
- La información recogida en el módulo de análisis y almacenada en la tabla de símbolos se emplea en la fase de síntesis para la generación del código objeto.
- Abarca la generación de código intermedio, y en el caso de los compiladores, la generación de código y la optimización del mismo.

Agrupamiento de fases

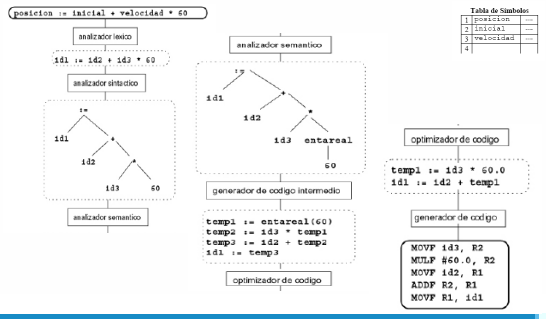
- Un compilador real no sigue la estructura descrita, pues sería poco eficiente.
- En la práctica, todas las funciones sobre la gramática se agrupan en un solo módulo (sintáctico + semántico + código intermedio) que analiza la sintaxis a la vez que la semántica y va generando código intermedio al mismo tiempo.
- Para unir la generación de código intermedio, se asocia a cada regla gramatical las instrucciones para generar el código intermedio.
- De esta forma se consigue un compilador más rápido y sencillo: **compilador dirigido por la sintaxis**.

Agrupamiento de fases

- **Front-end**: depende del lenguaje fuente, independiente del lenguaje objeto.
- **Back-end**: depende del lenguaje objeto y es independiente del lenguaje fuente.



Un primer vistazo al proceso



Tipos de compilador

- **Cruzado**: genera código en lenguaje objeto para una plataforma diferente de la que se está utilizando para compilar.
- De **una** o **varias pasadas**, según el número de veces que lea el programa fuente.
- **Autocompilador**: compilador escrito en el mismo lenguaje que va a compilar (el leguaje fuente y el de implementación coinciden)
- **Compilador con montador**: compila distintos módulos de forma independiente y después es capaz de enlazarlos.
- **Descompilador**: pasa de lenguaje objeto a fuente.

Aplicación de estas técnicas

- Diseño de un compilador (poco probable).
- Tratamiento de ficheros de texto con información estructurada.
- Procesadores de texto.
- Traducción de formatos de ficheros.
- Gestión de bases de datos:
  - En exploración y proceso de ficheros e interfaz de usuario.
- Procesamiento de lenguaje natural:
  - En análisis léxico y sintáctico.
- Cálculo simbólico.
  - Manejo simbólico de fórmulas.
- Reconocimiento de formas:
  - En detección de patrones de texto, reconocimiento automático de habla o la visión por computador.

Bibliografía básica

- *Compilers: Principles, Techniques, and Tools, 2nd Ed.*, A.V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, Addison Wesley, 2007

- *Compiladores: principios, técnicas y herramientas*. A.V. Aho, R. Sethi, J.D. Ullman. Addison-Wesley Iberoamerica. 1990.

Bibliografía complementaria

- K. C. Louden, *Compiler Construction: Principles and Practice*. Broks/Cole Publishing. 1997.
- Muchnick, S., Advanced Compiler Design Implementation. Morgan Kaufmann Pulishers, 1997.
- Wirth, N., Compiladores. Editorial Rueda,1990.