Compiladores vs Intérpretes

Compilador

En el caso de que el lenguaje fuente sea un lenguaje de programación de alto nivel y el objeto sea un lenguaje de bajo nivel (ensamblador o código de máquina), a dicho traductor se le denomina **compilador**. Un intérprete no genera un programa equivalente, sino que toma una sentencia del programa fuente en un lenguaje de alto nivel y la traduce al código equivalente y al mismo tiempo lo ejecuta.

Históricamente, con la escasez de memoria de los primeros ordenadores, se puso de moda el uso de intérpretes frente a los compiladores, ya que el programa fuente, sin traducir, y el intérprete juntos ocupaban una porción de memoria menor que la resultante de los compiladores. Por ello los primeros ordenadores personales iban siempre acompañados de un intérprete de BASIC (Spectrum, Commodore VIC-20, PC XT de IBM, etc.).

La *mejor información sobre los errores* por parte del compilador así como una *mayor velocidad de ejecución* del código resultante hizo que poco a poco se impusieran los compiladores. Hoy en día, y con el problema de la memoria prácticamente resuelto, se puede hablar de un gran predominio de los compiladores frente a los intérpretes, aunque intérpretes como los incluidos en los navegadores de Internet para interpretar el código JVM de Java son la gran excepción.

Ventajas

- Se compila una vez, se ejecuta n veces.
- El programa completo es verificado para que no haya errores sintácticos o semánticos.
- Se da más información sobre los errores.
- El archivo ejecutable es optimizado por el compilador para que se ejecute de manera más rápida.
- Los usuarios no tienen que ejecutar el programa en la misma máquina en la que fue realizado.

Intérprete

Los programas **interpretados** suelen ser *más lentos* que los **compilados**, pero *los* **intérpretes** son *más flexibles como entornos de programación y depuración*.

Comparando su actuación con la de un ser humano, un compilador equivale a un traductor profesional que, a partir de un texto, prepara otro independiente traducido a otra lengua, mientras que un intérprete corresponde al intérprete humano, que traduce en viva voz las palabras que oye, sin dejar constancia por escrito.

Ventajas

- Necesita menos memoria que un compilador.
- Podes encontrar los errores antes de terminar el programa y aprendes de ellos.
- El programa puede ser ejecutado antes de que sea finalizado entonces podes tener resultados parciales.
- Podes ir viendo los resultados y podes decidir si seguir o empezar de nuevo.
- Podes crear pequeñas partes de un proyecto y combinarlas después en un proyecto grande.

Un compilador no es un programa que funciona de manera aislada, sino que necesita de otros programas para conseguir su objetivo: obtener un programa ejecutable a partir de un programa fuente en un lenguaje de alto nivel. Algunos de esos programas son el *preprocesador*, el *linker*, el *depurador* y el *ensamblador*. El **preprocesador** se ocupa (dependiendo del lenguaje) de incluir ficheros, expandir macros, eliminar comentarios, y otras tareas similares. El ***linker*** se encarga de construir el fichero ejecutable añadiendo al fichero objeto generado por el compilador las cabeceras necesarias y las funciones de librería utilizadas por el programa fuente. El **depurador** permite, si el compilador ha generado adecuadamente el programa objeto, seguir paso a paso la ejecución de un programa. Finalmente, muchos compiladores, en vez de generar código objeto, generan un programa en lenguaje ensamblador que debe después convertirse en un ejecutable mediante un programa ensamblador.

Clasificación de Compiladores

El programa compilador traduce las instrucciones en un lenguaje de alto nivel a instrucciones que la computadora puede interpretar y ejecutar. **Para cada lenguaje de programación se requiere un compilador separado**. El compilador traduce todo el programa antes de ejecutarlo. Los compiladores son, entonces, programas de traducción insertados en la memoria por el sistema operativo para convertir programas de cómputo en pulsaciones electrónicas ejecutables (lenguaje de máquina). Los compiladores pueden ser de:

- Una sola pasada: examina el código fuente una vez, generando el código o programa objeto.
- Pasadas múltiples: requieren pasos intermedios para producir un código en otro lenguaje, y una pasada final para producir y optimizar el código producido durante los pasos anteriores.
- Optimación: lee un código fuente, lo analiza y descubre errores potenciales sin ejecutar el programa.
- Compiladores incrementales: generan un código objeto instrucción por instrucción (en vez de hacerlo para todo el programa) cuando el usuario teclea cada orden individual. El otro tipo de compiladores requiere que todos los enunciados o instrucciones se compilen conjuntamente.
- Ensamblador: el lenguaje fuente es lenguaje ensamblador y posee una estructura sencilla.
- Compilador cruzado: se genera código en lenguaje objeto para una máquina diferente de la que se está utilizando para compilar. Es perfectamente normal construir un compilador de Pascal que genere código para MS-DOS y que el compilador funcione en Linux y se haya escrito en C++.
- Compilador con montador: compilador que compila distintos módulos de forma independiente y después es capaz de enlazarlos.
- Autocompilador: compilador que está escrito en el mismo lenguaje que va a compilar. Evidentemente, no se puede ejecutar la primera vez. Sirve para hacer ampliaciones al lenguaje, mejorar el código generado, etc.
- Metacompilador: es sinónimo de compilador de compiladores y se refiere a un programa que recibe como entrada las especificaciones del lenguaje para el que se desea obtener un compilador y genera como salida el compilador para ese lenguaje. El desarrollo de los metacompiladores se encuentra con la dificultad de unir la generación de código con la parte de análisis. Lo que sí se han desarrollado son generadores de analizadores léxicos y sintácticos. Por ejemplo, los conocidos:
    - generador de analizadores léxicos LEX:
    - generador de YACC: analizadores sintácticos desarrollados para UNIX. Los inconvenientes que tienen son que los analizadores que generan no son muy eficientes.

• Descompilador: es un programa que acepta como entrada código máquina y lo traduce a un lenguaje de alto nivel, realizando el proceso inverso a la compilación.

Funciones de un compilador

A grandes rasgos un compilador es un programa que lee un programa escrito es un lenguaje, el lenguaje fuente, y lo traduce a un programa equivalente en otro lenguaje, el lenguaje objeto. Como parte importante de este proceso de traducción, el compilador informa a su usuario de la presencia de errores en el programa fuente.

A primera vista, la diversidad de compiladores puede parecer abrumadora. Hay miles de lenguajes fuente, desde los lenguajes de programación tradicionales, como FORTRAN o Pascal, hasta los lenguajes especializados que han surgido virtualmente en todas las áreas de aplicación de la informática. Los lenguajes objeto son igualmente variados; un lenguaje objeto puede ser otro lenguaje de programación o el lenguaje de máquina de cualquier computador entre un microprocesador y un supercomputador. A pesar de existir una aparente complejidad por la clasificación de los compiladores, como se vio en el tema anterior, las tareas básicas que debe realizar cualquier compilador son esencialmente las mismas. Al comprender tales tareas, se pueden construir compiladores para una gran diversidad de lenguajes fuente y máquinas objeto utilizando las mismas técnicas básicas.

Nuestro conocimiento sobre cómo organizar y escribir compiladores ha aumentado mucho desde que comenzaron a aparecerlos primeros compiladores a principios de los años cincuenta. Es difícil dar una fecha exacta de la aparición del primer compilador, porque en un principio gran parte del trabajo de experimentación y aplicación se realizó de manera independiente por varios grupos. Gran parte de los primeros trabajos de compilación estaba relacionada con la traducción de fórmulas aritméticas a código de máquina.

En la década de 1950, se consideró a los compiladores como programas notablemente difíciles de escribir. EL primer compilador de FORTRAN, por ejemplo, necesitó para su implantación de 18 años de trabajo en grupo (Backus y otros [1975]). Desde entonces, se han descubierto técnicas sistemáticas para manejar muchas de las importantes tareas que surgen en la compilación. También se han desarrollado buenos lenguajes de implantación, entornos de programación y herramientas de software. Con estos avances, puede hacerse un compilador real incluso como proyecto de estudio en un curso de un semestre sobre diseño sobre de compiladores.

Una clasificación muy importante de los lenguajes es la de hacer dos grupos en base a su funcionamiento, esto es considerarlos como intérpretes y compiladores, según se describe seguidamente.

Un lenguaje se dice que es un intérprete, por ejemplo los BASIC, cuando para ejecutar un programa el lenguaje ha de leer y traducir al lenguaje de la máquina las instrucciones una por una. Como es lógico el proceso se enlentece, por ejemplo si una operación está dentro de la estructura conocida como ciclo y este se repite 100 veces, el lenguaje tiene que traducirlo 100 veces al código de la máquina. No todo son desventajas, pues la parte buena de este tipo de lenguajes es que los errores se pueden corregir al momento y seguir facilmente la ejecución del programa, por lo cual son idóneos para aprender a programar, proceso en el que da lo mismo la lentitud.

Por contra un lenguaje se dice que es compilado, cuando el programa entero se traduce mediante el compilador de dicho lenguaje al lenguaje máquina correspondiente y el resultado se almacena de manera permanente en un archivo. De esta forma el programa se ejecutará de forma mucho más rápida que con un intérprete, sobre todo si hay estructuras que se repiten, caso de los ciclos. La principal desventaja es cuando se produce un error, que muchas veces se detecta en el momento de la ejecución , y la corrección no se puede hacer de inmediato, sino que hay que realizar todo el proceso de compilado desde el principio. Un ejemplo típico de lenguaje de este tipo el C ++, ampliamente usado en el desarrollo de programas.

Hay un lenguaje difundido en el ámbito de la enseñanza, es el Turbo Pascal, cuya ventaja es que aunque se trata de un compilador tiene un entorno de trabajo como si fuera un intérprete y cualquier error se puede corregir al momento, reanudándose la ejecución del programa de inmediato.

Un ejemplo en la vida real que visualiza la diferencia entre un intérprete y un compilador es el siguiente, supongamos que tenemos un libro escrito en una lengua distinta al castellano, hay dos procesos de acceder a su contenido cuando se necesite su uso, una es traducir en el momento de su empleo la parte del libro que se necesite, pero sin transcribirla a papel, sino simplemente mediante lectura traduciendo, esto sería el proceso de interpretado, mientras que la otra opción sería traducir el libro entero al castellano y dejar dicha versión escrita sobre papel, esto sería equivalente al compilado.

El proceso de compilado no es tan inmediato como parece, se describe seguidamente de forma muy esquematizada:

1. Se escribe el programa (conocido como programa fuente) mediante un editor de textos y se almacena en un fichero.
2. Este programa fuente es invocado por la primera etapa del compilador, que efectúa un análisis léxico, se puede considerar como una interpretación del programa fuente preparándolo para un tratamiento posterior con detalle. En esta etapa se ejecutan los tres procesos indicados seguidamente:

Adaptar el código fuente a un formato independiente de la forma en que se haya introducido en el sistema Eliminación de información redundante como espacios y comentarios

Tratar las palabras clave y los símbolos para su paso a símbolos clave, conocido como “tokens”

3. Análisis sintáctico es el paso siguiente, el compilador determina la estructura, y de alguna forma el significado del programa fuente. El conjunto del programa se analiza en bloques, que se descomponen en instrucciones y se procede a identificar los elementos individuales. Como la sintaxis está expresada mediante un conjunto de reglas, cada una indica como se construye una estructura del programa a partir de otras estructuras de menor entidad. El proceso mediante el cual el compilador aplica estas reglas es conocido como “parsing”.

Durante la compilación se genera gran cantidad de información, que se almacena en una estructura de datos conocida como diccionario o tabla de símbolos, en algún momento del proceso se necesitará la información guardada previamente. La mayor parte es información sobre variables, por ejemplo para X en el programa anterior, se tendría en el diccionario,

4. El paso siguiente es la generación de código, conocido como objeto Para ello se recorre el código intermedio generado y se busca cada uno de los “tokens” en el diccionario, lo que permite insertar las direcciones en el código máquina que se está generando.

5. El proceso siguiente es la optimización de código objeto generado, con lo que se consigue un programa más eficiente. Generalmente donde se consiguen los mejores resultados es en los bucles, cuyo objetivo es reducir al máximo el número de operaciones que se ejecutan en él.

6. Normalmente hay una etapa posterior conocida como “linkado “ en la que el o los módulos objetos generados previamente se unen entre sí y/o con otros módulos disponibles en librerías, para formar un fichero que contiene un programa ejecutable directamente desde el sistema operativo, sin necesidad de disponer del compilador correspondiente. Incluso se pueden unir programas escritos en lenguajes distintos si los módulos objeto creado se han estructurado de forma adecuada. En cualquier fase del proceso pueden detectarse errores que se podrán de manifiesto, implicando el volver hasta el programa fuente, efectuar las correcciones pertinentes y repetir todo el proceso, lo cual suele ser algo laborioso.

En el contexto de los lenguajes de programación, un intérprete es un programa que toma un programa fuente escrito en un lenguaje de alto nivel, lo analiza y lo ejecuta instrucción a instrucción bajo su control. En este caso, no se genera un programa equivalente en otro lenguaje de menor nivel, como ocurre con un compilador por lo que, si se desea repetir la ejecución del programa es preciso volver a traducirlo.

Así, un intérprete realizaría el siguiente ciclo

1. repetir
2. leer instrucción
3. ejecutar instrucción
4. hasta el fin

En un intérprete sólo hay que distinguir dos lenguajes diferentes, el de los programas de partida (LA), y el lenguaje en que está escrito el intérprete (LC).

Comparando su actuación con la de un ser humano, un compilador equivale a un traductor profesional que, a partir de un texto, prepara otro independiente traducido a otra lengua, mientras que un intérprete corresponde al intérprete humano, que traduce de viva voz las palabras que oye, sin dejar constancia por escrito.

Así, mientras un intérprete toma las instrucciones del programa fuente y las traduce y ejecuta a lenguaje máquina una a una, un compilador realiza la traducción completa del programa fuente a codigo máquina, sin ejecutarlo, siendo posteriormente cuando se ejecute el programa una vez compilado.

Ventajas del intérprete frente al compilador:

- El programa se puede ejecutar de inmediato, sin esperar a ser compilado.
- Puede ser interrumpido con facilidad.
- puede ser rápidamente modificado y ejecutado nuevamente.
- Resultan muy apropiados durante la fase de desarrollo de un programa, ya que la compilación no permite la ejecución paso a paso del programa y con ello impide la edición seguimiento y depuración del programa.

Desventajas del intérprete frente al compilador:

- La ejecución es más lenta, pues cada intrucción debe ser traducida a código máquina tantas veces como sea ejecutada.
- No son adecuados en la fase de explotación del programa ya que el proceso de interpretación se ha de repetir cada vez que se ejecuta el programa, mientras que con la compilación, una vez obtenido el programa en leguaje máquina éste puede ser ejecutado sin necesidad de compilarlo de nuevo.

Un intérprete es un traductor de lenguaje, igual que un compilador, pero difiere de éste en que ejecuta el programa fuente inmediatamente, en vez de generar un código objeto que se ejecuta después de que se completa la traducción. En principio, cualquier lenguaje de programación se puede interpretar o compilar, pero se puede preferir un intérprete a un compilador dependiendo del lenguaje que se esté usando y de la situación en la cual se presenta la traducción.

Los intérpretes se utilizan con frecuencia en situaciones relacionadas con la enseñanza o con el desarrollo de software, donde los programas son probablemente traducidos y vueltos a traducir muchas veces. Por otra parte, es preferible usar un compilador si lo que importa es la velocidad de ejecución, ya que el código de objeto compilado es siempre más rápido que el código fuente interpretado, en ocasiones hasta por un factor de 10 o más.

No obstante, los interpretes comparten muchas de sus operaciones con los compiladores, y ahí pueden incluso ser traductores híbridos, de manera que quedan en alguna parte entre los intérpretes y los compiladores. Existen lenguajes cuyos traductores se idearon como intérpretes y otros como compiladores. No obstante, para un lenguaje dado, pueden existir tanto compiladores como intérpretes.

En resumen…

Los COMPILADORES e INTÉRPRETES son software capaz de traducir de un lenguaje de alto nivel al lenguaje ensamblador específico de una máquina. Los primeros toman todo el programa en lenguaje de alto nivel, lo pasan a lenguaje ensamblador y luego lo ejecutan. Los últimos toman instrucción por instrucción, la traducen y la van ejecutando.

Compilador:

- Es un programa que traduce los programas escritos en lenguajes de alto nivel a lenguaje máquina.
- Los programas escritos en lenguajes de alto nivel se llaman programas fuente y
- El programa traducido se llama programa objeto.
- El compilador traduce sentencia a sentencia el programa fuente.

Algunos lenguajes compiladores típicos son:

- C
- C++
- Pascal
- FORTRAN
- COBOL

Intérprete:

- Es un traductor que toma un programa fuente, lo traduce y a continuación lo ejecuta.

Los programas interpretes clásicos son:

- BASIC
- QBASIC
- QUICKBASIC
- VISUALBASIC
- SMALLTALK
- JAVA