Fases de la Traducción y Errores

7.3

¿Qué son las **fases de traducción**?

Un programa consiste en una o más unidades de traducción almacenadas en archivos. Es traducido en varias fases. Las primeras fases hacen transformaciones léxicas de bajo nivel, ejecutan directivas introducidas con líneas que principian con el carácter #, y realizan macro definiciones y expansiones. Cuando el **preprocesamiento está completo**, el programa se ha reducido a una secuencia de **componentes léxicos**.

Diagrama

El contenido generado por IA puede ser incorrecto.

Lo primero que le ocurre a un fichero .c es el preprocesado. En este paso se sustituyen todas las macros y se eliminan los comentarios. El resultado, es un fichero de código (. i).

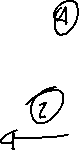
El segundo paso es la compilación, en el código **C** preprocesado se convierte en código ensamblador, que es un fichero (. s).

El tercer paso es el ensamblado del código ensamblador, lo que lo convierte en código máquina. Un fichero de código máquina o fichero objeto (. o).

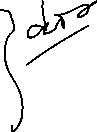
Preprocesamiento

Las instrucciones en la etapa de preprocesamiento generalmente comienzan con #, por lo que los programas que escribimos como #include, #define, etc.se completan en esta etapa, a menudo lo usamos cuando escribimos programas Para algunos archivos de encabezado stdio o iostream, no son solo una oración simple, son algunas bibliotecas que se han escrito, podemos citarlas directamente aquí, por lo que la etapa de preprocesamiento abrirá todos los archivos de encabezado que cite Para insertar en nuestro propio programa. La segunda es que la etapa de preprocesamiento reemplazará todas las macros en nuestro programa. A menudo definimos una macro al comienzo del programa, y ​​el reemplazo de la definición de macro se realiza en la etapa de preprocesamiento. La tercera tarea es que a menudo escribimos algunos comentarios cuando escribimos programas. Por lo tanto, el programa eliminará la compilación que escribimos durante la etapa de preprocesamiento. La máquina no puede ver los comentarios que escribimos. El cuarto es nuestra compilación condicional. A menudo escribimos #ifdef. En este momento, nuestra máquina no verá la parte que no cumple con las condiciones y no entrará en la etapa de compilación.

es una serie de [instrucciones](https://es.wikipedia.org/wiki/Instrucci%C3%B3n_(inform%C3%A1tica)) que se almacenan para que se puedan ejecutar de manera secuencial mediante una sola llamada u orden de ejecución



Después de hablar de esto, debemos mencionar nuestro gcc. Presentamos **vim** antes. Se puede decir que Vim es un **editor de programas**. Los programas de Linux están separados. Vim no dice que puedas compilar tu programa después de escribirlo. Usted ejecuta, estas tareas son completadas por gcc, y gcc puede ejecutar los cuatro pasos mencionados anteriormente un paso a la vez, podemos ver qué cambios se realizan en cada paso del programa. Aquí usamos gcc para ver qué cambios ocurren en cada paso del programa.



Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.



La función de -E aquí es detener el programa después de que se completa el preprocesamiento, y luego -o se refiere al archivo de destino, lo que significa que se llama al archivo generado después del procesamiento -E de hello\_original.c. El archivo i ha sido preprocesado Programa fuente en lenguaje C. En este momento, podemos ver que hay un archivo.i en nuestra carpeta anterior en ls y podemos abrirlo.

Compilador

En la segunda etapa, la tarea principal es dejar que el compilador verifique si su programa tiene algunos errores gramaticales. Cuando escribimos el programa en VS, también lo compilaremos para ver si nuestro programa tiene errores. O **advertirlo y ejecutarlo nuevamente**, pero no es un buen hábito dejar que el programa se ejecute directamente para ahorrar tiempo. Cuando su programa no tenga problemas, la compilación acercará la programación de su programa al lenguaje ensamblador del lenguaje máquina.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.



En este momento, compilaremos el archivo hello\_original.i mediante la instrucción -S significa que nuestro archivo solo se compilará y no se ensamblará para generar código ensamblador.

Assembler

La tercera etapa es la etapa de ensamblaje, esta **etapa es convertir el código ensamblador generado en la segunda etapa en nuestro archivo ejecutable**, que es convertir nuestro lenguaje ensamblador en un lenguaje máquina que puede ser ejecutado por nuestra máquina. La etapa por la que debe pasar el programa, porque nuestro lenguaje C o ensamblador o todo tipo de máquinas de lenguaje no pueden entenderlo.



Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.



-C significa dejar que nuestro programa ejecute la tercera etapa para generar un lenguaje máquina que la máquina pueda entender. En este momento, podemos ver el archivo .o

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

-C (mayúscula), que en realidad es para preservar comentarios durante el preprocesamiento, el archivo resultante no es un objeto válido sino un ejecutable malformado. Por eso el enlazador (ld) se queja de que no puede usar un archivo ejecutable como entrada.



Texto

El contenido generado por IA puede ser incorrecto.



[Fases del compilador con ejemplo: proceso y pasos de compilación](https://www.guru99.com/es/compiler-design-phases-of-compiler.html)

[Las cuatro etapas del compilador gcc: preprocesamiento, compilación, ensamblaje, vinculación - programador clic](https://programmerclick.com/article/38531616876/)

7.3.1

<https://www.reddit.com/r/learnprogramming/comments/18m6lp/x86_asm_confusion_over_gcc_generated_assemblygas/?rdt=54289>

[x86 Assembly Language Programming](https://cs.lmu.edu/~ray/notes/x86assembly/)

[pcasm-book.pdf](https://pacman128.github.io/static/pcasm-book.pdf)

<https://en.cppreference.com/w/c/io/fprintf>

[Assembly](https://rodrigocadiz.github.io/SD-Notebooks/SD-D-Assembly.html)