Practica 0 Compiladores

Angel Sandoval Mendoza

Agosto 2024

3 inciso

Comparación de programa.i con stdio.h

Al comparar los archivos programa.i con stdio.h, nos damos cuenta de que al ejecutar el código, programa.i incorporó funciones, definiciones y otras características que estaban incluidas en stdio.h, además de aplicar algunos valores directamente.

¿Qué hace cpp?

El preprocesador cpp realiza algunos procesos para que el código fuente pueda ser ejecutado por el procesador, como incorporar las definiciones que están en el código, eliminar comentarios, evaluar funciones, etc. Finalmente, genera un código intermedio.

4 inciso

-Wall

Esta instrucción habilita todas las alertas del compilador, como variables que no se utilizan, conversiones de tipos problemáticas, etc. Los desarrolladores la utilizan para lograr un código más limpio y optimizado.

-S

Esta instrucción indica al compilador que se detenga cuando se haya generado el código en ensamblador.

Archivo de salida y cuál es su extensión

El archivo de salida contiene instrucciones en ensamblador basadas en el código fuente original y en las operaciones en ensamblador del equipo utilizado. La extensión del archivo generado en este apartado es ".s".

5 inciso

¿Qué esperamos en el archivo .o?

Al ensamblar el archivo ".s", esperamos que estas instrucciones se conviertan en código máquina o instrucciones booleanas.

Contenido de .o

Este archivo contiene operaciones en binario que se envían directamente al CPU para su ejecución. Esto es importante porque mejora la eficiencia del programa y permite combinar varios archivos de código fuente en uno solo.

as

Se invoca al GNU Assembler, el cual se encarga de convertir el código ensamblador en un archivo objeto.

8 inciso

Al ejecutar el comando:

```
ld -dynamic-linker /lib64/ld-linux-x86-64.so.2 -pie
/usr/lib/x86_64-linux-gnu/Scrt1.o
/usr/lib/x86_64-linux-gnu/crti.o
/usr/lib/gcc/x86_64-linux-gnu/9/crtbeginS.o
-L/usr/lib/gcc/x86_64-linux-gnu/9
-L/usr/lib/x86_64-linux-gnu
-L/lib/x86_64-linux-gnu
-L/usr/lib
/home/angelsm/Documents/programa.o
-lgcc --as-needed -lgcc_s --no-as-needed -lc
-lgcc --as-needed -lgcc_s --no-as-needed
/usr/lib/gcc/x86_64-linux-gnu/9/crtendS.o
/usr/lib/x86_64-linux-gnu/crtn.o
-o programa
```

Tuve que modificar algunas cosas del código que se nos dio en la práctica. Agregué las direcciones de los archivos, quité algunos espacios y añadí algunos símbolos para que se pudiera ejecutar.

Nos devuelve un archivo ejecutable que contiene todo lo necesario para llevar a cabo la ejecución del algoritmo. Investigando un poco, la configuración de esta indicación hace más seguro el uso del ejecutable.

10 inciso

Al ejecutar el programa, parece que el valor asignado a PI hace que el cálculo del área sea más preciso. Sin embargo, en cuanto a estructura, no parece haber un cambio significativo.

11 inciso

Se ha creado un programa en el lenguaje C como fue solicitado en la práctica correspondiente, en este se utilizaron las directivas hacia el preprocesador de C, que serían las siguientes:

#include <stdio.h>

Nos permite agregar el contenido de stdio.h, que en este caso nos ayuda con las funciones de entrada y salida requeridas para el programa.

#define

Nos ayuda a definir macros, que son básicamente sustituciones de texto; cuando el preprocesador encuentre el nombre asignado, cambiará el macro por su valor asignado anteriormente. En el programa, nos ayuda a definir los valores, variables y las frases que se utilizarán.

#undef

Nos ayuda a eliminar las macros que se definen y no se pueden volver a utilizar a menos que se vuelvan a definir. En el programa en cuestión, solo nos ayuda a dar un ejemplo de su utilidad.

#warning

Manda una señal de alerta al momento de compilación, pero no afecta a la ejecución del programa. En este caso, solo da una advertencia de que es humor.

#if, #elif, #else, #endif

Este conjunto es para compilación condicional. #if nos permite verificar si la condición dada es verdadera, #elif y #else nos ayudan a agregar otras condiciones, y #endif se utiliza para terminar las condiciones. En el programa, nos ayuda a definir el tipo de dato y cuáles aceptar.

#error

Es usado para detener la compilación si se cumplen algunas condiciones. En el código, nos ayuda a que se ingrese el tipo correcto de dato.

Conclusiones

Esta práctica me ha ayudado a comprender todo el trabajo que hace el compilador de C, desde la preprocesación hasta el ensamblado y enlazado del código, viendo cada parte del proceso paso a paso, brindándome herramientas para generar un código más eficiente, correcto y dándome un conocimiento más profundo de cómo funciona algo tan complejo como sería el proceso de C.