Universidad Nacional Autónoma de México
Facultad de Ciencias
Compiladores
Práctica 1: Sistema de procesamiento de Lenguaje
Profesor: Adrián Ulises Mercado Martínez

Sistema de procesamiento de Lenguaje

Objetivo:

Que el alumno identifique el funcionamiento de los distintos programas que permiten llevar acabo el proceso de traducción.

Introducción

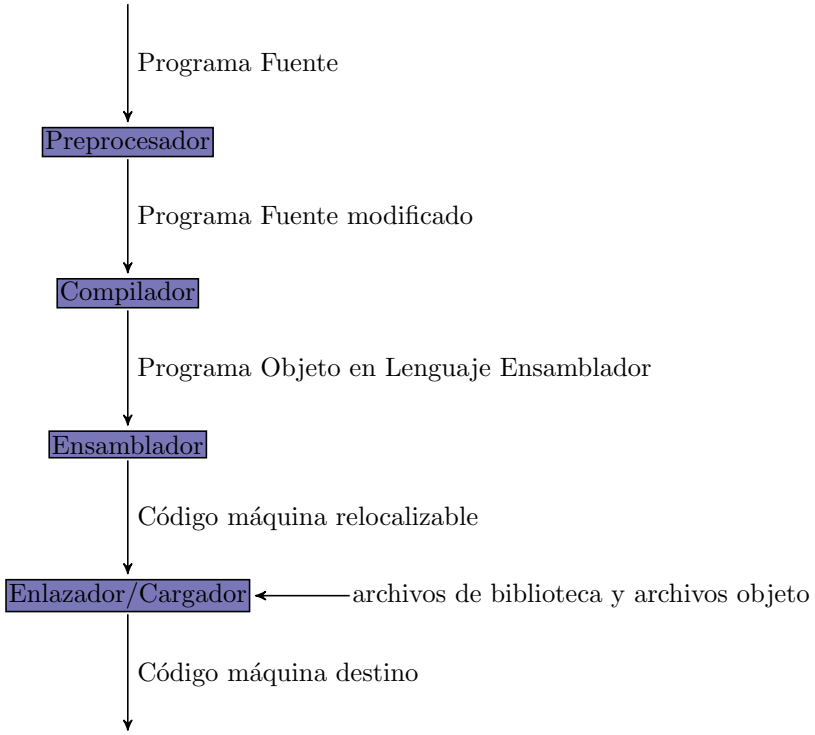
Un compilador es un programa que se encarga de hacer la traducción de un programa fuente escrito en lenguaje de alto nivel a un programa escrito en lenguaje objeto que por lo general es un lenguaje de bajo nivel.

Para realizar esta traducción el compilador se auxilia de otros programas como lo son el preprocesador, que se encarga de recolectar el programa escrito en módulos en archivos separados, expandir fragmentos de código abreviados de uso frecuentes, llamados macros.

El programa modificado por el preprocesador ingresa al compilador, éste producirá el programa destino escrito en lenguaje ensamblador, que a continuación es procesado por un ensamblador que genera el código de máquina.

Una vez que el programa ha sido ensamblado, es necesario vincular los archivos de código máquina con otros archivos objeto y de biblioteca para que se produzca el código ejecutable.

Finalmente el cargador lleva el archivo objeto a la memoria para su ejecución.



Desarrollo:

1. Deberá tener instalado el compilador gcc y trabajar en un ámbiente Linux.

2. Escriba el siguiente programa en lenguaje c. y nombrelo programa.c

```c
#include <stdio.h>

//#define PI 3.1415926535897

#ifdef PI
#define area(r) (PI * r * r)
#else
#define area(r) (3.1416 * r * r)
#endif


/**
 * Compiladores 2019-2
 *
 */
int main(void){
  printf("Hola Mundo!\n");
  float mi_area = area(3); //soy un comentario... hasta donde llegare?
  printf("Resultado: %f\n",mi_area);
  return 0;
}
```

3. Use el siguiente comando: **cpp programa.c programa.i**
   Revise el contenido de *programa.i* y conteste lo siguiente:

   (a) ¿Qué ocurre cuando se invoca el comando cpp con esos argumentos?
   (b) ¿Qué similitudes encuentra entre los archivos *programa.c* y *programa.i*?
   (c) ¿Qué pasa con las macros y los comentarios del código fuente original en *programa.i*?
   (d) Compare el contenido de *programa.i* con el de *stdio.h* e indique de forma general las similitudes entre ambos archivos.
   (e) ¿A qué etapa corresponde este proceso?

4. Ejecute la siguiente instrucción: **gcc -Wall -S programa.i**

   (a) ¿Para qué sirve la opción -*Wall*?
   (b) ¿Qué le indica a gcc la opción -*S*?
   (c) ¿Qué contiene el archivo de salida y cuál es su extensión?
   (d) ¿A qué etapa corresponde este comando?

5. Ejecute la siguiente instrucción: **as programa.s -o programa.o**

   (a) Antes de revisarlo, indique cuál es su hipótesis sobre lo que debe contener el archivo con extensión *.o*.
   (b) Diga de forma general qué contiene el archivo *programa.o* y por qué se visualiza de esa manera.
   (c) ¿Qué programa se invoca con *as*?
   (d) ¿A qué etapa corresponde la llamada a este programa?

6. Encuentre la ruta de los siguientes archivos en el equipo de trabajo:

   - Scrt1.o
   - crti.o
   - crtbeginS.o
   - crtendS.o
   - crtn.o

7. Ejecute el siguiente comando, sustituyendo las rutas que encontró en el paso anterior:

   ld -dynamic-linker /lib64/**ld-linux-x86-64.so.2** -pie **/ruta/para/Scrt1.o**
   /ruta/para/crti.o /ruta/para/crtbeginS.o
   -L/usr/lib/gcc/**x86_64-pc-linux-gnu/7.3.1** -L/usr/lib -L/lib -L/usr/lib
   -L/usr/lib programa.o -lgcc –as-needed -lgcc_s –no-as-needed -lc -lgcc –as-needed
   -lgcc_s –no-as-needed **/ruta/para/crtendS.o /ruta/para/crtn.o**
   -o **ejecutable**

(a) En caso de que el comando ld mande errores, investigue como enlazar un programa utilizando el comando ld. Y proponga una posible solución para llevar a cabo este proceso con éxito.

(b) En caso de ser exitoso ¿Qué se genero al ejecutar el comando anterior?

(c) Describa qué ocurre si omitimos alguno de los archivos con terminación *.o* en el comando.

8. Quite el comentario de la macro **#define PI** en el código fuente original y conteste lo siguiente:

   (a) ¿Al ejecutar el comando del rubro 3. cambia en algo la salida alojada en *programa.i*? (Puedes llamar el mismo comando con un segundo argumento diferente al de la primera vez para facilitar su comparación. P.ej. **cpp programa.c programa_v2.i**)

   (b) ¿Cambia en algo la ejecución final?

9. Escriba el código fuente de un segundo programa en lenguaje C en el que agregue 4 directivas del preprocesador de C (*cpp*)[1]. Las directivas elegidas deben jugar algún papel en el significado del programa, ser distintas entre sí y diferentes de las utilizadas en el primer programa (aunque no están prohibidas si las requieren). Explique su utilidad general y su función en particular para su programa.

10. Escriba sus resultados al ejecutar el programa compilado y sus conclusiones personales.

[1] Pueden consultar la lista de directivas en su documentación en línea: CPP - Index of directives.
O bien, revisar la entrada para este preprocesador en la herramienta *man* en *Linux*: `$ man cpp`