Compiladores (Clave: 1764)

Tarea: 01 - Proceso de Compilación (En Linux)

Arellanes Conde Esteban Méndez Galicia Axel Gael

Contents

1 Objetivo

El objetivo de esta tarea es comprender y analizar las distintas etapas del proceso de compilación en un entorno Linux, utilizando el compilador *'gcc'*. Para ello, se debe escribir un programa en lenguaje C, compilarlo con la biblioteca matemática *('-lm')*, y examinar los diferentes archivos generados en cada fase. A través de esta tarea, comprenderemos el funcionamiento interno de un compilador y cómo el código fuente se transforma en un programa ejecutable:

1. **Preprocesamiento:** Analizar la expansión de macros y directivas de inclusión.
2. **Compilación:** Convertir el código fuente en ensamblador y examinar su estructura.
3. **Generación de código objeto:** Observar el binario intermedio antes del enlazado.
4. **Enlazado:** Crear el ejecutable final uniendo los archivos objeto con las bibliotecas necesarias.
5. **Exploración del binario y ensamblador:** Inspeccionar el código ensamblador y el contenido del ejecutable.

2 Programa: Escribir el código fuente

```c
#include <stdio.h>
#include <math.h>

int main() {
    double x = 9.0;
    double y = sqrt(x);
    printf("La raiz cuadrada de %.2f es %.2f\n", x, y);
    return 0;
}
```

Creación de un programa (*funMat.c*) en C

```
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# cat funMat.c
#include <stdio.h>
#include <math.h>

int main() {
    double x = 9.0;
    double y = sqrt(x);
    printf("La raiz cuadrada de %.2f es %.2f\n", x, y);
    return 0;
}
```

Figure 1: Creación de un programa (*funMat.c*) en C

En esta primera etapa creamos un archivo de nombre *funMat.c* con un editor como *nano*, *vi*, *vim*, *neovim*, entre otros:

3 Preprocesamiento (cpp):

Esta etapa expande macros y directivas *#include*, *#define*, etc. Se puede observar el resultado con los siguientes comandos:

```
$ gcc -E funMat.c -o funMat.i
```

```
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# gcc -E funMat.c -o funMat.i
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# man gcc
```

Figure 2: Preprocesamiento

Esto genera el archivo *funMat.i*, que contiene el código fuente con las bibliotecas expandidas; a su vez la bandera *-E* de acuerdo al manual de gcc nos indica lo siguiente:

-E Stop after the preprocessing stage; do not run the compiler proper. The output is in the form of preprocessed source code, which is sent to the standard output.

Para visualizarlo:

```
$ cat funMat.i | head -n 15
$ less funMat.i
```

```
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# cat funMat.i | head -n 15
# 0 "funMat.c"
# 0 "<built-in>"
# 0 "<command-line>"
# 1 "/usr/include/stdc-predef.h" 1 3 4
# 0 "<command-line>" 2
# 1 "funMat.c"
# 1 "/usr/include/stdio.h" 1 3 4
# 28 "/usr/include/stdio.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 1 3 4
# 33 "/usr/include/x86_64-linux-gnu/bits/libc-header-start.h" 3 4
# 1 "/usr/include/features.h" 1 3 4
# 394 "/usr/include/features.h" 3 4
# 1 "/usr/include/features-time64.h" 1 3 4
# 20 "/usr/include/features-time64.h" 3 4
# 1 "/usr/include/x86_64-linux-gnu/bits/wordsize.h" 1 3 4
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01#
```

Figure 3: Preprocesamiento

4 Compilación a ensamblador (cc1)

Esto convierte el código fuente en ensamblador. Ejecutando los comandos:

```
$ gcc -S funMat.c -o funMat.s -lm
```

El archivo funMat.s contiene el código en ensamblador, la bandera -S según el manual nos dicta lo siguiente: *Stop after the stage of compilation proper; do not assemble. The output is in the form of an assembler code file for each non-assembler input file specified. By default, the assembler file name for a source file is made by replacing the suffix .c, .i, etc., with .s. Input files that don't require compilation are ignored.* Y la bandera -lm es para utilizar la librería math. Se puede observar con:

```
$ cat funMat.s | head -n 15
$ less funMat.s
```

```
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# gcc -n funMat.c -o funMat.o -lm
/usr/bin/ld: cannot find -lgcc_s: No such file or directory
/usr/bin/ld: cannot find -lgcc_s: No such file or directory
collect2: error: ld returned 1 exit status
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# gcc -S funMat.c -o funMat.s -lm
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# cat funMat.s | head -n 15
        .file   "funMat.c"
        .text
        .section        .rodata
        .align 8
.LC1:
        .string "La ra\303\255z cuadrada de %.2f es %.2f\n"
        .text
        .globl  main
        .type   main, @function
main:
.LFB0:
        .cfi_startproc
        endbr64
        pushq   %rbp
        .cfi_def_cfa_offset 16
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# less funMat.s
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01#
```

Figure 4: Enter Caption

5 Compilación a código máquina (objeto) (as)

Convierte el código ensamblador en código máquina, generando un archivo objeto:

```
$ gcc -c funMat.c -o funMat.o -lm
```

El archivo funMat.o está en binario. Por lo tanto una forma de inspeccionarlo es mediante:

```
$ objdump -d funMat.o | less
```

6 Enlazado (Linking) y generación del ejecutable (ld)

Aquí se unen los archivos objeto y bibliotecas necesarias para generar el ejecutable:

```
$ gcc funMat.o -o funMat -lm
```

Aquí se unen los archivos objeto y bibliotecas necesarias para generar el ejecutable:

./funMat

```
0000670   00116   00000   00000   00000   00000   00000   00000   00000
0000680   00001   00000   00000   00000   00000   00000   00000   00000
0000690
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# gcc funMat.o -o funMat -lm
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# ./funMat
La raíz cuadrada de 9.00 es 3.00
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01#
```

Figure 5: Enlace a el archivo objeto

7 Ver el binario del ejecutable

Si se quiere inspeccionar el binario o ver la versión ensamblador del binario final, hay que ejecutar los siguientes comandos para observar un volcado en hexadecimal:

```
$ hexdump -C funMat | less
$ objdump -d funMat | less
```

```
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# gcc -c funMat.c -o funMat.o -lm
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# objdump -d funMat.o | less
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# objdump -d funMat.o

funMat.o:     file format elf64-x86-64


Disassembly of section .text:

0000000000000000 <main>:
   0:   f3 0f 1e fa             endbr64
   4:   55                      push   %rbp
   5:   48 89 e5                mov    %rsp,%rbp
   8:   48 83 ec 10             sub    $0x10,%rsp
   c:   f2 0f 10 05 00 00 00    movsd  0x0(%rip),%xmm0        # 14 <main+0x14>
  13:   00
  14:   f2 0f 11 45 f0          movsd  %xmm0,-0x10(%rbp)
  19:   48 8b 45 f0             mov    -0x10(%rbp),%rax
  1d:   66 48 0f 6e c0          movq   %rax,%xmm0
  22:   e8 00 00 00 00          call   27 <main+0x27>
  27:   66 48 0f 7e c0          movq   %xmm0,%rax
  2c:   48 89 45 f8             mov    %rax,-0x8(%rbp)
  30:   f2 0f 10 45 f8          movsd  -0x8(%rbp),%xmm0
  35:   48 8b 45 f0             mov    -0x10(%rbp),%rax
  39:   66 0f 28 c8             movapd %xmm0,%xmm1
  3d:   66 48 0f 6e c0          movq   %rax,%xmm0
  42:   48 8d 05 00 00 00 00    lea    0x0(%rip),%rax        # 49 <main+0x49>
  49:   48 89 c7                mov    %rax,%rdi
  4c:   b8 02 00 00 00          mov    $0x2,%eax
  51:   e8 00 00 00 00          call   56 <main+0x56>
  56:   b8 00 00 00 00          mov    $0x0,%eax
  5b:   c9                      leave
  5c:   c3                      ret
root@pc-bd-ace:/home/bante/Documents/Compiladores/Tarea_01# hexdump -d funMat.o
0000000   17791   17996   00258   00001   00000   00000   00000   00000
0000010   00001   00062   00001   00000   00000   00000   00000   00000
0000020   00000   00000   00000   00000   00784   00000   00000   00000
0000030   00000   00000   00064   00000   00000   00064   00014   00013
0000040   04083   64030   18517   58761   33608   04332   04082   01296
0000050   00000   00000   04082   17681   18672   17803   26252   03912
```

Figure 6: Objdump & Hexdump

8 References

Bibliographic References and Web Sites consulted:

* [1] **A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman, Compilers: Principles, Techniques, and Tools, 2nd ed. Boston, MA, USA: Pearson, 2006.**