



**Universidad Nacional Autónoma de México**

**Facultad de Ingeniería**

**Rangel López Victor Manuel**

**Ejercicio Practico 1**

**Compiladores**

**Grupo 1**

**2023-1**

**Ing. Adrián Ulises Mercado Marínez**

## 1. Deberá tener instalado el compilador gcc o trabajar en un ambiente Linux.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19044.1889]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\victo>gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=C:/Program Files/CodeBlocks/MinGW/bin/./libexec/gcc/x86_64-w64-mingw32/8.1.0/lto-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../../src/gcc-8.1.0/configure --host=x86_64-w64-mingw32 --build=x86_64-w64-mingw32 --target=x86_64-w64-mingw32 --prefix=/mingw64 --with-sysroot=/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64 --enable-shared --enable-static --disable-multilib --enable-languages=c,c++,fortran,lto --enable-libstdcxx-time=yes --enable-threads-posix --enable-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking-release --enable-fully-dynamic-string --enable-version-specific-runtime-libs --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disable-rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=nocona --with-tune=core2 --with-libiconv --with-system-zlib --with-gmp=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpfr=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpc=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-isl=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-pkgversion='x86_64-posix-seh-rev0, Built by MinGW-W64 project' --with-bugurl=https://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' CPPFLAGS='-I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' LDFLAGS='-pipe -fno-ident -L/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/lib -L/c/mingw810/prerequisites/x86_64-zlib-static/lib -L/c/mingw810/prerequisites/x86_64-w64-mingw32-static/lib'
Thread model: posix
gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)
```

## 2. Escriba el siguiente código en un archivo llamado programa.c

```
Start here X programa.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  // #define PI 3.1415926535897
4  #ifdef PI
5  #define area(r) (PI*r*r)
6  #else
7  #define area(r) (3.1416*r*r)
8  #endif
9  /*
10   * Este es un programa de prueba, para verificar
11   * el funcionamiento del sistema de procesamiento de lenguaje
12   */
13  int main ( void ) {
14      printf ( "Hola Mundo!\n" ) ; // Funcion para imprimir hola mundo
15      float mi_area = area ( 3 ) ; // soy un comentario . . .
16      printf ( "Resultado : %f \n" , mi_area ) ;
17      return 0;
18  }
19
```

### 3. Usar el siguiente comando: `cpp programa.c > programa.i`

```
C:\Users\victo\Downloads>cpp programa.c > programa.i
C:\Users\victo\Downloads>
```

(a) Busque los archivos `.h` y revise su contenido

(b) Compare el contenido de `programa.i` con el de `stdio.h` y `stdlib.h`, indique de forma general las similitudes entre los archivos `.h` y el `.i`

(c) Observe lo que ocurrió con los comentarios y las directivas de preprocesador.

`programa.i`

```
programa: Bloc de notas
Archivo Edición Formato Ver Ayuda
free(_Memory);
}

}
}
# 209 "C:/Program Files/CodeBlocks/MinGW/x86_64-w64-mingw32/include/malloc.h" 3
#pragma pack(pop)
# 742 "C:/Program Files/CodeBlocks/MinGW/x86_64-w64-mingw32/include/stdlib.h" 2 3
# 3 "programa.c" 2
# 13 "programa.c"

# 13 "programa.c"
int main ( void ) {
printf ( "Hola Mundo!\n" );
float mi_area = (3.1416*3*3);
printf( "Resultado : %f \n" , mi_area );
return 0;
}
```

`Stdio.h`

```
stdio: Bloc de notas
Archivo Edición Formato Ver Ayuda
/**
 * This file has no copyright assigned and is placed in the Public Domain.
 * This file is part of the mingw-w64 runtime package.
 * No warranty is given; refer to the file DISCLAIMER.PD within this package.
 */
#ifndef _INC_STDIO
#define _INC_STDIO

#include <crtdefs.h>

#include <_mingw_print_push.h>

#pragma pack(push,_CRT_PACKING)

#ifdef __cplusplus
extern "C" {
#endif

#define BUFSIZ 512
#define _NFILE _NSTREAM_
#define _NSTREAM_ 512
#define _IOB_ENTRIES 20
#define EOF (-1)
```

`Stdlib.h`

```
stdlib: Bloc de notas
Archivo Edición Formato Ver Ayuda
#ifndef _CRT_ERRNO_DEFINED
#define _CRT_ERRNO_DEFINED
_CRTIMP extern int *_cdecl _errno(void);
#define errno (*_errno())
errno_t __cdecl _set_errno(int _Value);
errno_t __cdecl _get_errno(int *_Value);
#endif
_CRTIMP unsigned long *_cdecl _doserrno(void);
#define _doserrno (*_doserrno())
errno_t __cdecl _set_doserrno(unsigned long _Value);
errno_t __cdecl _get_doserrno(unsigned long *_Value);
#ifdef _MSVCRT_
extern char *_sys_errlist[];
extern int _sys_nerr;
#else
#ifdef _MSVCRT_VERSION_ >= 0x1400
_CRTIMP char **_cdecl _sys_errlist(void);
_CRTIMP int *_cdecl _sys_nerr(void);
#define _sys_nerr (*_sys_nerr())
#define _sys_errlist (_sys_errlist())
#else
extern __declspec(dllimport) char *_sys_errlist[1];
extern __declspec(dllimport) int _sys_nerr;
#endif /* _MSVCRT_VERSION_ < 0x1400 */
```

#### 4. Ejecute la siguiente instruccion: gcc -S programa.i

```
C:\Windows\system32\cmd.exe  
  
C:\Users\victo\Downloads>gcc -S programa.i
```

(a) ¿Que opción se agrega para que gcc muestre todas las advertencias durante este proceso?

-Wall.

Ejemplo:

```
$ gcc -Wall -o main_es main_es.c
```

(b) ¿Que le indica a gcc la opción -S?

La opción '-S' dicta a gcc a convertir el código fuente C a lenguaje ensamblador sin crear un fichero objeto.

Ejemplo:

```
$ gcc -Wall -S hola.i
```

(c) ¿Que contiene el archivo de salida y cuál es su extensión?

El lenguaje ensamblador resultante es almacenado en el fichero 'programa.s'. El lenguaje ensamblador contiene una llamada a la función externa puts, una versión simple de printf para cadenas que no contienen caracteres formateados.

```
programa: Bloc de notas  
Archivo Edición Formato Ver Ayuda  
|  
.file "programa.c"  
.text  
.def __main; .scl 2; .type 32; .endef  
.section .rdata,"dr"  
.LC0:  
.ascii "Hola Mundo!\0"  
.LC2:  
.ascii "Resultado : %f \12\0"  
.text  
.globl main  
.def main; .scl 2; .type 32; .endef  
.seh_proc main  
main:  
pushq %rbp  
.seh_pushreg %rbp  
movq %rsp, %rbp  
.seh_setframe %rbp, 0  
subq $48, %rsp  
.seh_stackalloc 48  
.seh_endprologue  
call __main  
leaq .LC0(%rip), %rcx  
call puts  
movss .LC1(%rip), %xmm0  
movss %xmm0, -4(%rbp)  
cvtss2sd -4(%rbp), %xmm0  
movq %xmm0, %rax  
movq %rax, %rdx  
movq %rdx, %xmm1  
movq %rax, %rdx  
leaq .LC2(%rip), %rcx  
call printf  
movl $0, %eax  
addq $48, %rsp  
popq %rbp  
ret
```

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

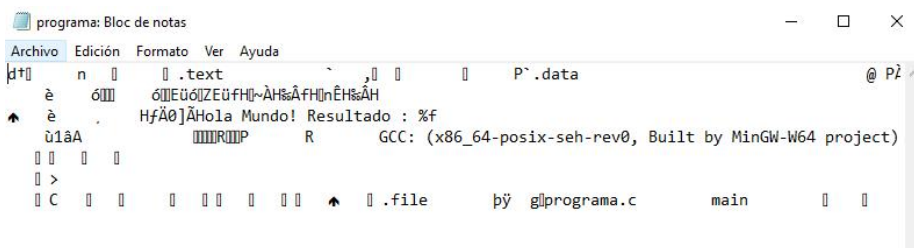
```
C:\Users\victo\Downloads>as programa.s -o programa.o  
C:\Users\victo\Downloads>
```

### (a) ¿Formule una hipótesis sobre el contenido del archivo.o?

Lo que ocurre es que el ensamblador convierte de lenguaje ensamblador a código máquina creando así un fichero objeto.

### (b) ¿Que representa el contenido del programa .o de acuerdo con el diagrama de un sistema de procesamiento de lenguaje?

Es el enlace de ficheros objeto para crear un ejecutable.



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

- `Scrt1.o` o `crt1.o`
- `crti.o`
- `crtbeginS.o` o `ctrbegin.o`
- `crtendS.o` o `ctrend.o`
- `crti.o`

En este punto de la práctica, logré encontrar la carpeta en la que se encontraban los archivos .o, sin embargo, por alguna razón que desconozco, no se encontraban todos los archivos mostrados en la lista.

Archivos de programa > CodeBlocks > MinGW > x86\_64-w64-mingw32 > lib

| Nombre     | Fecha de modificación  | Tipo      |
|------------|------------------------|-----------|
| txtmode    | 11/05/2018 09:50 p. m. | Archivo O |
| gcrt2      | 11/05/2018 09:50 p. m. | Archivo O |
| gcrt1      | 11/05/2018 09:50 p. m. | Archivo O |
| gcrt0      | 11/05/2018 09:50 p. m. | Archivo O |
| dllcrt2    | 11/05/2018 09:50 p. m. | Archivo O |
| dllcrt1    | 11/05/2018 09:50 p. m. | Archivo O |
| crtend     | 11/05/2018 09:50 p. m. | Archivo O |
| crtbegin   | 11/05/2018 09:50 p. m. | Archivo O |
| crt2u      | 11/05/2018 09:50 p. m. | Archivo O |
| crt2       | 11/05/2018 09:50 p. m. | Archivo O |
| crt1u      | 11/05/2018 09:50 p. m. | Archivo O |
| crt1       | 11/05/2018 09:50 p. m. | Archivo O |
| CRT_noglob | 11/05/2018 09:50 p. m. | Archivo O |
| CRT_glob   | 11/05/2018 09:50 p. m. | Archivo O |
| CRT_fp10   | 11/05/2018 09:50 p. m. | Archivo O |
| CRT_fp8    | 11/05/2018 09:50 p. m. | Archivo O |
| binmode    | 11/05/2018 09:50 p. m. | Archivo 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
```

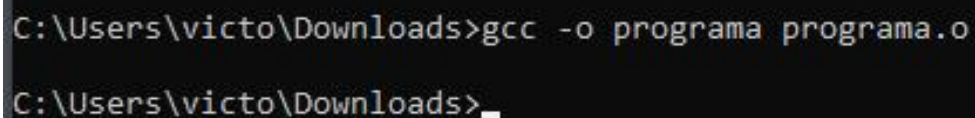
O bien

```
ld -o programa -dynamic-linker /lib64/ld-linux-x86-64.so.2 -pie /usr/lib/Scrt1.o /ruta/para/crti.o /ruta/para/crtbeginS.o programa.o -lc /ruta/para/crtendS.o /ruta/para/crtn.o
```

**(a) En caso de que el comando ld mande errores, investigue como enlazar un programa utilizando el comando ld.**

El propósito de esta práctica era conocer el proceso de compilación de una forma manual, sin embargo, al no encontrar todas las rutas anteriores, el paso de enlazamiento no lo pude realizar.

Utilizando el siguiente comando, se realiza el enlazamiento, así como todos los demás pasos anteriores. Esta es una forma más sencilla y más utilizada, sin embargo como dije antes; está fuera del propósito de la práctica.



```
C:\Users\victo\Downloads>gcc -o programa programa.o
C:\Users\victo\Downloads>_
```

**(b) ¿Que se generó al ejecutar el comando anterior?**

En la etapa de enlace, se reúnen uno o más módulos en código objeto con el código existente en las bibliotecas.

8. Una vez que se enlazo el código máquina relocable, podemos ejecutar el programa con la siguiente institución en la terminal: `./programa`

En Windows para correr un programa se utiliza el comando “start programa” en lugar de “`./programa`”.

```
C:\Users\victo\Downloads>start programa

C:\Users\victo\Downloads> C:\Users\victo\Downloads\programa.exe

Hola Mundo!
Resultado : 28.274401

Process returned 0 (0x0)   execution time : 0.093 s
Press any key to continue.
```

9. Quite el comentario de la macro `#define PI`

(a) Genere nuevamente el archivo.i. De preferencia asigne un nuevo nombre.

```
progra: Bloc de notas
Archivo Edición Formato Ver Ayuda
free(_Memory);
}

}
}
# 209 "C:/Program Files/CodeBlocks/MinGW/x86_64-w64-mingw32/include/malloc.h" 3
#pragma pack(pop)
# 742 "C:/Program Files/CodeBlocks/MinGW/x86_64-w64-mingw32/include/stdlib.h" 2 3
# 3 "progra.c" 2
# 13 "progra.c"

# 13 "progra.c"
int main ( void ) {
printf ( "Hola Mundo!\n" ) ;
float mi_area = (3.1415926535897*3*3) ;
printf( "Resultado : %f \n" , mi_area ) ;
return 0;
}
```

(b) ¿Cambio algo en la ejecución final?

```
C:\Users\victo\Downloads>cpp programa.c > programa.i

C:\Users\victo\Downloads>cpp progra.c > progra.i

C:\Users\victo\Downloads>gcc -S progra.i

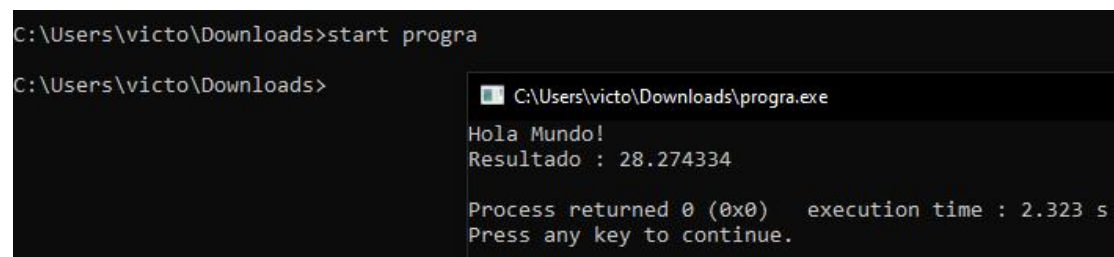
C:\Users\victo\Downloads>as progra.s -o progra.o

C:\Users\victo\Downloads>gcc -o progra progra.o

C:\Users\victo\Downloads>
```

El cambio se produce cuando se realiza el calculo del area, ya que en el primero se toma un valor de 3.1416. En cambio, al quitar el comentario, se define que PI va a tomar un valor diferente, en el cual se utilizaran más decimales para hacer la operación.

Al final, al ejecutar el programa los valores son similares, sin embargo, si difieren uno del otro por unos cuantos decimales.



```
C:\Users\victo\Downloads>start progra
C:\Users\victo\Downloads>
C:\Users\victo\Downloads\progra.exe
Hola Mundo!
Resultado : 28.274334
Process returned 0 (0x0)   execution time : 2.323 s
Press any key to continue.
```

## 10. Escribe un segundo programa en lenguaje C

En el agregue 4 directivas del preprocesador de C (cpp)\*. Las directivas elegidas deben jugar algún papel en el significado del programa, ser distintas entre sí y ser diferentes de las utilizadas en el primer programa (aunque no están prohibidas, si las requieren).

(a) Explique la utilidad general de las directivas usadas y su función en particular para su programa.

- **#undef**

Se usa #undef para quitar una definición de nombre de macro que se haya definido previamente. El formato general es:

#undef <nombre de macro>

El uso principal de #undef es permitir localizar los nombres de macros sólo en las secciones de código que los necesiten.

- **#line**

La directiva #line número ``cadena" informa al preprocesador cual es el número siguiente de la línea de entrada. Cadena es opcional y nombra la siguiente línea de entrada. Esto es usado frecuentemente cuando son traducidos otros lenguajes a C. Por ejemplo, los mensajes de error



producidos por el compilador de C pueden referenciar el nombre del archivo y la línea de la fuente original en vez de los archivos intermedios de C.

- **#if**

La directiva '#if' le permite probar el valor de una expresión aritmética, en lugar de la mera existencia de una macro. Su sintaxis es

```
#if expression

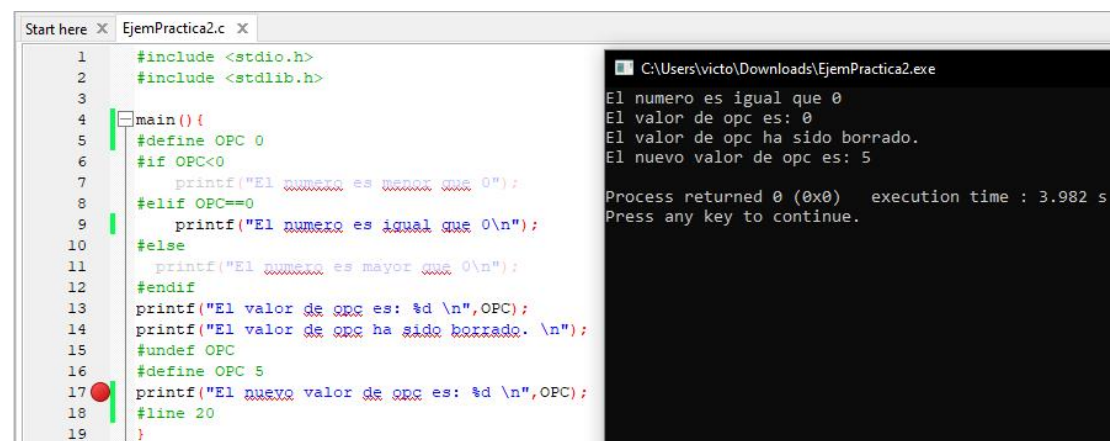
controlled text

#endif /* expression */
```

- **#elif**

'#elif' significa "si no". Como '#else', va en medio de un grupo condicional y lo subdivide; no requiere un '#endif' coincidente propio. Al igual que '#if', la directiva '#elif' incluye una expresión para probar. El texto que sigue a '#elif' se procesa solo si la condición '#if' original falló y la condición '#elif' tiene éxito.

```
#if X == 1
...
#elif X == 2
...
#else /* X != 2 and X != 1*/
...
#endif /* X != 2 and X != 1*/
```



```
Start here X EjemPractica2.c X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 main(){
5     #define OPC 0
6     #if OPC<0
7         printf("El numero es menor que 0\n");
8     #elif OPC==0
9         printf("El numero es igual que 0\n");
10    #else
11        printf("El numero es mayor que 0\n");
12    #endif
13    printf("El valor de opc es: %d \n",OPC);
14    printf("El valor de opc ha sido borrado. \n");
15    #undef OPC
16    #define OPC 5
17    printf("El nuevo valor de opc es: %d \n",OPC);
18    #line 20
19 }
```

```
C:\Users\victo\Downloads\EjemPractica2.exe
El numero es igual que 0
El valor de opc es: 0
El valor de opc ha sido borrado.
El nuevo valor de opc es: 5

Process returned 0 (0x0)   execution time : 3.982 s
Press any key to continue.
```

## **11. Escriba sus resultados y conclusiones.**

Al realizar esta practica, hicimos el proceso de compilación de un programa en lenguaje C de una manera manual. Dejamos a un lado la opción que involucra simplemente presionar el botón que dice “copilar y correr”, para en esta ocasión hacerlo nosotros mismos desde la terminal y darnos cuenta de que en lo que ocurre en cada una de las cuatro etapas de compilación de un programa (preprocesamiento, compilación, ensamblado y enlazado).

En primer lugar nos encontramos con el preprocesado donde se interpretan las directivas al preprocesador. En seguida se encuentra la compilación, que transforma el código C en el lenguaje ensamblador. El ensamblado transforma el programa escrito en lenguaje ensamblador a código objeto, un archivo binario en lenguaje de máquina ejecutable por el procesador. Finalmente en el enlazado las funciones incluidas en el código, se encuentran ya compiladas y ensambladas en bibliotecas existentes en el sistema.

A mi parecer la parte más difícil de toda la practica fue el enlazado, ya que para ello se necesitaba de ciertas rutas de archivos .o, sin embargo, por alguna razón no logré encontrarlas. Esto no fue una limitante, por el contrario, en el afán de intentar solucionar ese problema, pude investigar y encontrar el comando que reúne todos los pasos en uno solo (`gcc -o programa programa.c`).

Hoy en día el programar y compilar nuestros códigos va siendo cada vez más fácil, sin embargo (como todo en la vida), tiene un origen. Esta práctica nos regresó al pasado, para así comprender todo lo que hay detrás, todos y cada uno de los pasos que se requieren para la compilación de un programa y darnos cuenta que no solo es presionar un boto con las opción que diga “compilar y correr”