

## Taller 2: GCC

### Investigación.

Para comprender mejor GCC, realice una pequeña búsqueda para responder las siguientes preguntas:

#### 1. ¿Qué es GCC?

GNU Compiler Collection es un compilador integrado del proyecto GNU para C, C++, Objective C y Fortran. Es capaz de recibir un programa fuente en cualquiera de estos lenguajes y generar un programa ejecutable binario en el lenguaje de la máquina donde se ha de ejecutar.

<https://www.scayle.es/manual/es/hpc/software-instalado/gcc>

#### 2. ¿Cuáles son las 4 etapas de compilación?

Preprocesamiento, compilación, ensamblaje y vinculación.

[https://aurea.es/wp-content/uploads/compilacionlinux\\_gcc.pdf](https://aurea.es/wp-content/uploads/compilacionlinux_gcc.pdf)

#### 3. ¿Qué comando debería utilizar para generar el código en ensamblador de un archivo fuente, por ejemplo, calculadora.c?

En general se tiene el siguiente caso:

```
gcc programa.c -S
```

En este caso:

```
gcc calculadora.c -S
```

[https://aurea.es/wp-content/uploads/compilacionlinux\\_gcc.pdf](https://aurea.es/wp-content/uploads/compilacionlinux_gcc.pdf)

#### 4. ¿Cuál es la diferencia entre una biblioteca estática y una dinámica?

**Librerías Estáticas.** Se enlazan al compilar, quedan "dentro" del ejecutable final. En Windows tienen la extensión .lib. En Linux tienen la extensión .a.

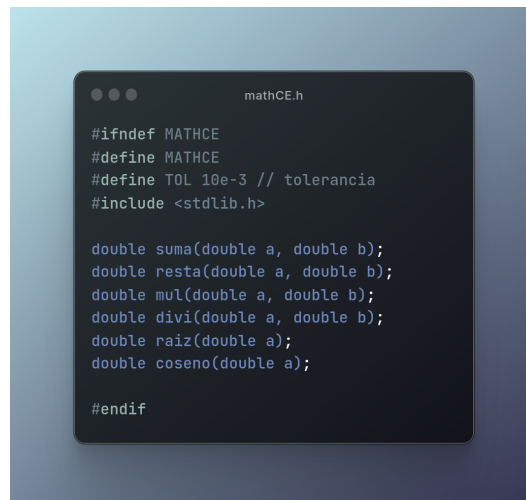
**Librerías Dinámicas.** Se enlazan al ejecutar, el sistema operativo debe encontrarlas al ejecutar el programa. Si una aplicación se instaló bien, el sistema operativo no debe tener problema para encontrarla. En Windows tienen la extensión .dll (en general en c:\windows\system32 y sin número de versión, de ahí el llamado dll hell). En Linux tienen la extensión .so (normalmente en /usr/lib o /usr/local/lib); dicho sistema operativo actualiza su listado con un sudo ldconfig).

<https://asrob.uc3m.es/tutoriales/software/programming/libs.html>

## Práctica.

Debe crear una biblioteca, en lenguaje C, la cual ofrecerá seis funciones matemáticas: suma, resta, multiplicación, división, raíz cuadrada y coseno (por aproximación). Para el desarrollo de la biblioteca, tome en cuenta los siguientes aspectos:

**Debe crear un archivo de cabecera biblioteca.h (los nombres de archivos son genéricos, puede cambiarlos a conveniencia), por ejemplo, que contenga únicamente la definición de las funciones a utilizar y las variables globales que requiera. Debe crear además el archivo biblioteca.c, donde desarrolle cada una de las funciones.**

A screenshot of a code editor window titled 'mathCE.h'. The code defines a header file for a C library. It starts with an ifndef guard, followed by a define for MATHCE and a define for TOL (10e-3). It includes <stdlib.h>. Then it declares six functions: suma, resta, mul, divi, raiz, and coseno, all taking double arguments and returning double. The file ends with an endif guard.

```
mathCE.h

#ifndef MATHCE
#define MATHCE
#define TOL 10e-3 // tolerancia
#include <stdlib.h>

double suma(double a, double b);
double resta(double a, double b);
double mul(double a, double b);
double divi(double a, double b);
double raiz(double a);
double coseno(double a);

#endif
```

Figura 1. Archivo .h

```
mathCE.c

#include "mathCE.h"

double fabs_t(double a){
    if( a < 0.0){
        return -1*a;
    } else return a;
}

double suma(double a, double b)
{
    return a + b;
}

double resta(double a, double b)
{
    return a - b;
}

double mul(double a, double b)
{
    return a * b;
}

double divi(double a, double b)
{
    // Error si el cociente es 0, ya que no se puede dividir
    if (b == 0.0)
        exit(-1);
    return a / b;
}

double raiz(double x)
{
    float t_ant, t_act;
    t_act = 1;
    do
    {
        t_ant = t_act;
        t_act = (t_ant + x / t_ant) / 2;
    } while (fabs_t(t_act - t_ant) >= TOL);
    return t_act;
}

double coseno(double x)
{
    int i, signo;
    float cos, ult;
    i = 1;
    cos = ult = 1;
    signo = -1;
    while (fabs_t(ult) > TOL)
    {
        ult = ult * x * x / i / (i + 1);
        cos += (signo * ult);
        signo *= -1;
        i += 2;
    }
    return cos;
}
```

Figura 2. Archivo .c

La biblioteca deberá ser creada tanto estática como dinámicamente, es decir, al finalizar el ejercicio deberá tener dos archivos: `libbiblioteca.a` y `libbiblioteca.so`

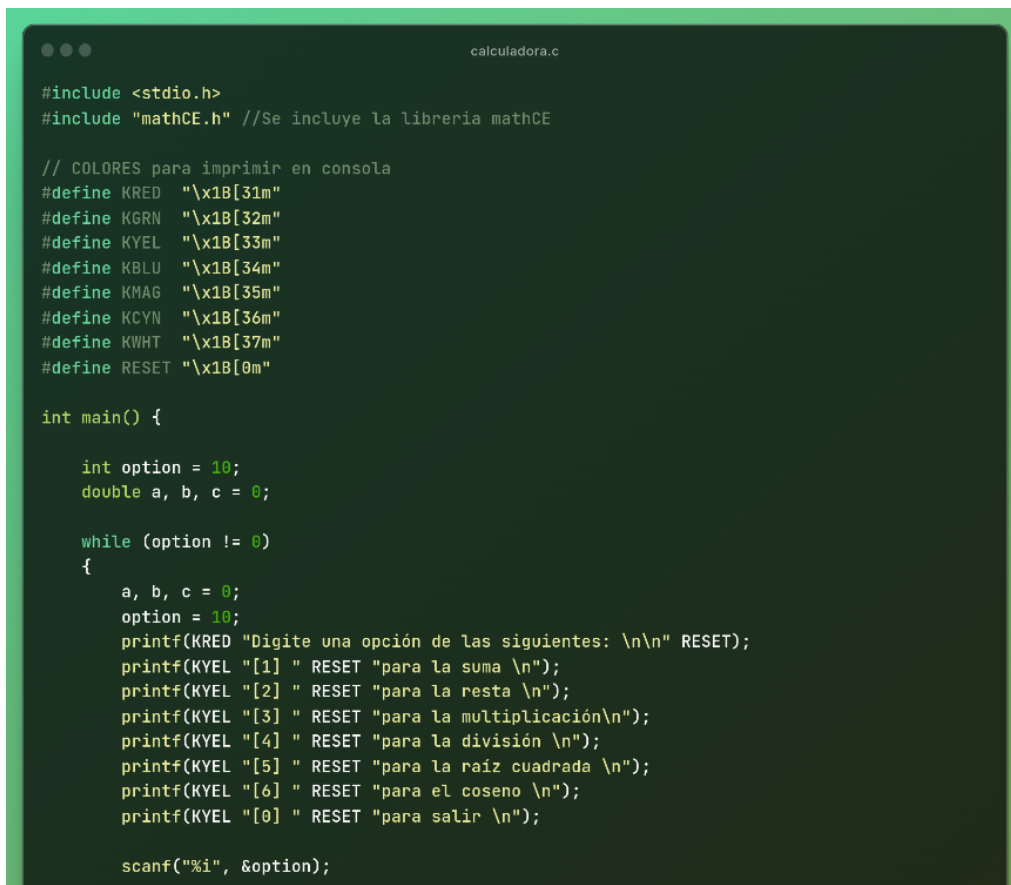
Para la estática se ejecutan los siguientes comandos:

```
gcc -c mathCE.c -o libMathCE.o
ar rcs libMathCE.a libMathCE.o
gcc -static calculadora.c -L. -l MathCE -o calc_static
```

Para la dinámica se ejecutan los siguientes comandos:

```
gcc -c -fPIC mathCE.c -o mathCE.o
gcc -shared -Wl,-soname,libMathCE.so.1 -o libMathCE.so.1.0 mathCE.o
gcc calculadora.c -o calc_dynamic -L. -lMathCE
```

Por último, debe crear un archivo `calculadora.c` y su correspondiente ejecutable, para verificar el correcto funcionamiento de ambas bibliotecas.



```
calculadora.c

#include <stdio.h>
#include "mathCE.h" //Se incluye la libreria mathCE

// COLORES para imprimir en consola
#define KRED  "\x1B[31m"
#define KGRN  "\x1B[32m"
#define KYEL  "\x1B[33m"
#define KBLU  "\x1B[34m"
#define KMAG  "\x1B[35m"
#define KCYN  "\x1B[36m"
#define KWHT  "\x1B[37m"
#define RESET "\x1B[0m"

int main() {

    int option = 10;
    double a, b, c = 0;

    while (option != 0)
    {
        a, b, c = 0;
        option = 10;
        printf(KRED "Digite una opción de las siguientes: \n\n" RESET);
        printf(KYEL "[1] " RESET "para la suma \n");
        printf(KYEL "[2] " RESET "para la resta \n");
        printf(KYEL "[3] " RESET "para la multiplicación\n");
        printf(KYEL "[4] " RESET "para la división \n");
        printf(KYEL "[5] " RESET "para la raíz cuadrada \n");
        printf(KYEL "[6] " RESET "para el coseno \n");
        printf(KYEL "[0] " RESET "para salir \n");

        scanf("%i", &option);
    }
}
```

Figura 3. Archivo `calculadora.c` parte 1.

```

switch (option)
{
case 1:
    printf(KGRN "Digite el primer número: \n" RESET);
    scanf("%lf", &a);
    printf(KGRN "Digite el segundo número: \n" RESET);
    scanf("%lf", &b);
    printf( KMAG "[SUMA] ---> " RESET "%.3lf + %.3lf, resultado ---> %.3f\n\n", a, b,
suma(a, b));
    continue;
case 2:
    printf(KGRN "Digite el primer número: \n" RESET);
    scanf("%lf", &a);
    printf(KGRN "Digite el segundo número: \n\n" RESET);
    scanf("%lf", &b);
    printf( KBLU "[RESTA] ---> " RESET "%.3lf - %.3lf, resultado ---> %.3lf\n\n", a, b
, resta(a, b));
    continue;
case 3:
    printf(KGRN "Digite el primer número: \n" RESET);
    scanf("%lf", &a);
    printf(KGRN "Digite el segundo número: \n\n" RESET);
    scanf("%lf", &b);
    printf( KRED "[MULTIPLICACION] ---> " RESET "%.3lf * %.3lf, resultado --->
%.3lf\n\n", a, b , mul(a, b));
    continue;
case 4:
    printf(KGRN "Digite el primer número: \n" RESET);
    scanf("%lf", &a);
    printf(KGRN "Digite el segundo número: \n\n" RESET);
    scanf("%lf", &b);
    printf( KYEL "[DIVISION] ---> " RESET "%.3lf / %.3lf, resultado ---> %.3lf\n\n",
a, b, divi(a, b));
    continue;
case 5:
    printf(KGRN "Digite el número a aplicar la raíz cuadrada: \n\n" RESET);
    scanf("%lf", &a);
    printf(KCYN "[RAIZ] ---> " RESET "%.3lf, resultado ---> %.3f\n\n", a, raiz(a));
    continue;
case 6:
    printf(KGRN "Digite el número a aplicar el coseno (en grados): \n\n" RESET);
    scanf("%lf", &a);
    printf( KGRN "[COSENO] ---> " RESET "cos(%3lf), resultado ---> %.3f\n\n", a,
coseno(a));
    continue;
case 0:
    break;

default:
    continue;
}
}

return 0;
}

```

Figura 4. Archivo calculadora.c parte 2.

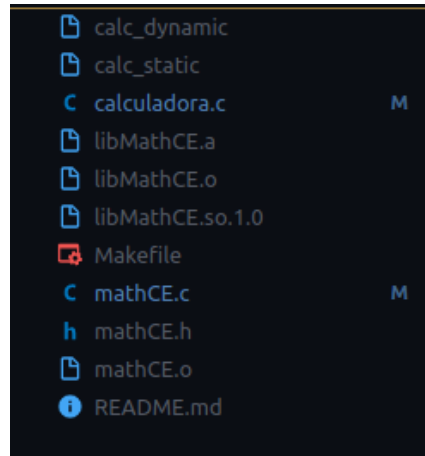


Figura 5. Archivos obtenidos al final del ejercicio.

Para facilitar la ejecución de comandos se crea un Makefile. Entonces para probar la librería estática solo se ejecuta el comando “make static” y para la dinámica se ejecuta “make dynamic”. Una vez hecho esto se obtendrán dos ejecutables los cuales podrá usar a conveniencia, ya que su funcionamiento es el mismo.

```

samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-2$ make static
gcc -c mathCE.c -o libMathCE.o
ar rcs libMathCE.a libMathCE.o
gcc -static calculadora.c -L. -l MathCE -o calc_static
samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-2$ make dynamic
gcc -c mathCE.c -o libMathCE.o
ar rcs libMathCE.a libMathCE.o
gcc -c -fPIC mathCE.c -o mathCE.o
gcc -shared -Wl,-soname,libMathCE.so.1 -o libMathCE.so.1.0 mathCE.o
gcc calculadora.c -o calc_dynamic -L. -lMathCE
samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-2$

```

Figura 6. Uso de los comandos del Makefile.

```

samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-2$ ./calc_static
Digite una opción de las siguientes:

[1] para la suma
[2] para la resta
[3] para la multiplicación
[4] para la división
[5] para la raíz cuadrada
[6] para el coseno
[0] para salir
1
Digite el primer número:
15
Digite el segundo número:
16
[SUMA] ---> 15.000 + 16.000, resultado ---> 31.000

```

Figura 7. Ejecución de la calculadora con la librería estática.

```
○ samastua@samastua-Inspiron-15-3567:~/Documents/TEC/Embebidos/Git/Taller-2$ ./calc_dynamic
Digite una opción de las siguientes:

[1] para la suma
[2] para la resta
[3] para la multiplicación
[4] para la división
[5] para la raíz cuadrada
[6] para el coseno
[0] para salir
3
Digite el primer número:
6
Digite el segundo número:
8
[MULTIPLICACION] ---> 6.000 * 8.000, resultado ---> 48.000
```

Figura 8. Ejecución de la calculadora con la librería dinámica.