Nombre de la práctica	CICLOS DE RELOJ			No.	6
Asignatura:	MÉTODOS NUMÉRICOS	Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES	Duración de la práctica (Hrs)	

I. **Competencia(s) específica(s):** * * * * * * * *
II. **Lugar de realización de la práctica (laboratorio, taller, aula u otro):**
Aula
III. **Material empleado:**
Visual Studio Code
IV. Desarrollo de la práctica:

Ciclos de reloj

Los ciclos de reloj, ciclos por segundo o frecuencia, es un termino que hace referencia a la velocidad del procesador incorporado en la CPU del ordenador, y se mide en Megahercios o Gigahercios, el numero de operaciones que puede realizar por segundo.

La unidad básica, el Hercio (hertz), mide el numero de instrucciones que el ordenador puede ejecutar por segundo (desde que comienza un ciclo hasta que vuelve a comenzar). Un Hercio es una instrucción por segundo, un Megahercio mil operaciones por segundo y un Gigahercio mil millones.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4  #include <sys/time.h>
5
6
7  double dif_tiempos(struct timeval *tiempoIn, struct timeval *tiempoFin){
8      return
9      (double)(tiempoIn->tv_sec +(double)tiempoIn->tv_usec/1000000)-
10     (double)(tiempoFin->tv_sec +(double)tiempoFin->tv_usec/1000000);
11 };
12
13 int main(int argc, char *argv[]){
14     struct timeval tiempo_inicial, tiempo_final;
15     double segundos;
16     gettimeofday(&tiempo_inicial, NULL);
17     system("libreoffice");
18     //system("libreoffice --writer");
19     gettimeofday(&tiempo_final,NULL);
20     segundos=dif_tiempos(&tiempo_final,&tiempo_inicial);
21     printf("%.16g milisegundos\n",segundos*1000.0);
22     return 0;
23 }
```

Se importan las bibliotecas necesarias, de las cuales se obtendrán las funciones con las que se trabajara.

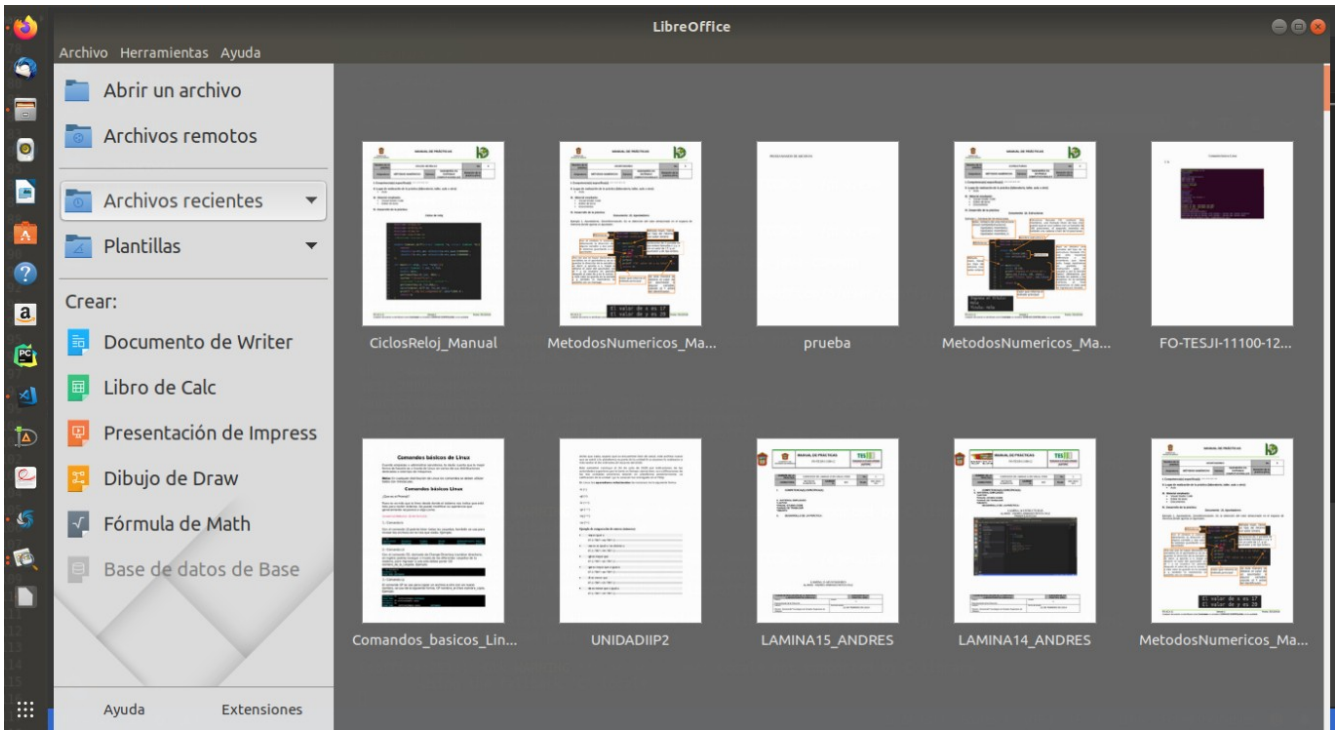
Para medir de forma precisa el tiempo que tarda en ejecutarse cierta operación, en este caso, abrir el programa de libreoffice, se debe usar la función gettimeofday, que se encuentra en la biblioteca “sys/time.h”.
Esta función ofrece teóricamente una precisión de microsegundos (0,001 milisegundos).
La sintaxis de la función es la siguiente:
Int gettimeofday (struct timeval*tp, NULL);
Siendo timeval un registro con dos campos: int tv_sec, int tv_usec, que indican los segundos y microsegundos, respectivamente.

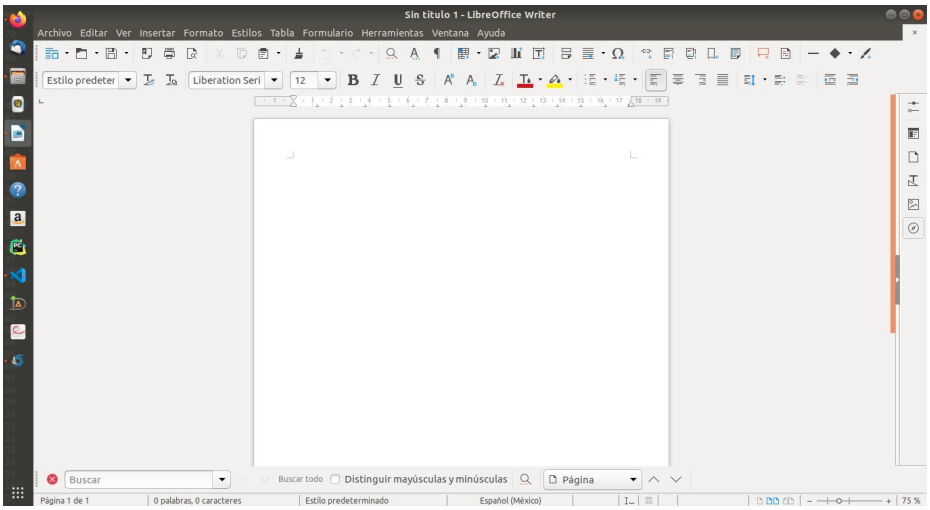
A partir del método principal se inicia la ejecución del programa, en el tenemos dos argumentos, uno de tipo entero llamado argc y el otro un apuntador de tipo char.
Después se encuentran declarados dos variables que serán del tipo de la estructura timeval que ya fue mencionada anteriormente, una llamada tiempo_inicial y tiempo_final que denotan el tiempo de inicio y fin respectivamente, de igual manera declaramos a la variable segundos de tipo flotante (double).
Enseguida de ello, indicamos el tiempo de inicio, con ayuda de la función gettimeofday, de igual manera, el tiempo final, pero dentro de estas dos lineas de código debemos especificar las instrucciones o el proceso que se realizara, es decir, aquel proceso del cual vamos a medir el tiempo de ejecución.

system() es una función del lenguaje de programación C incluida en su biblioteca estándar, dentro de la cabecera <stdlib.h>. Sirve para ejecutar subprocesos o comandos del sistema operativo.
Para ello indicamos dentro de esta función el programa que deseamos ejecutar (“libreoffice”).
Con esa simple instrucción podremos abrir libreoffice, hay dos maneras de hacerlo, una es abrir un documento o abrir el programa como tal. Las dos formas se muestran a continuación:

```
system(“libreoffice”);
 system("libreoffice --writer");
```

Después nos encontramos con la llamada la función timeval_diff, en la que le pasaremos como valores a los parámetros los tiempos de inicio y fin, para obtener los segundos que se tardo en abrir el programa, enseguida, guardamos el retorno de esa función en la variable secs, lo multiplicamos por 1000, ya que el valor se dará en milisegundos y se muestra en pantalla.
Al ejecutar el programa, libre office iniciara





Para abrir el programa de libreoffice usando apuntadores, en este caso, se hace uso del mismo código ya que de igual manera queremos medir los ciclos de reloj, así que lo único que añadiríamos seria el puntero para hacer la llamada al programa, en primer lugar declaramos al apuntador de tipo char, ya que este contendrá el nombre del programa al cual se va a apuntar, después dentro de la función system, haremos la llamada a ese apuntador, el cual se hará cargo de abrir libreoffice, y la función del resto del código sera la misma que el código anterior, con la única diferencia es que al ejecutar el programa, el segundo código tendrá menor cantidad de milisegundos que el primero, de ahi la importancia de trabajar con punterosm ya que estos reduccen los ciclos de reloj.

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <sys/time.h>


double dif_tiempos(struct timeval *tiempoIn, struct timeval *tiempoFin){
    return
    (double)(tiempoIn->tv_sec +(double)tiempoIn->tv_usec/1000000)-
    (double)(tiempoFin->tv_sec +(double)tiempoFin->tv_usec/1000000);
};

int main(int argc, char *argv[]){
    char *nombre="libreoffice";
    struct timeval tiempo_inicial, tiempo_final;
    double segundos;
    gettimeofday(&tiempo_inicial, NULL);
    system(nombre);
    system('"'+nombre+'"');
    //system("libreoffice --writer");
    gettimeofday(&tiempo_final,NULL);
    segundos=dif_tiempos(&tiempo_final,&tiempo_inicial);
    printf("%.16g milisegundos\n",segundos*1000.0);
    return 0;
}
```

```
4215.367078781128 milisegundos
```

V. Conclusiones:

Para terminar puedo decir que los ciclos de reloj nos permiten saber el desempeño de nuestro procesador, en el caso anterior la velocidad a la que puede ejecutar el programa, así mismo, cabe destacar que el uso de punteros reduce los antes mencionados, ademas que proporciona mejor desempeño en la ejecución de cualquier programa.
Esto tiene una mejor optimización y rendimiento, es decir, es mas eficaz la implementación de punteros.