# Práctica 2 - Programación con Pthreads

**Consideraciones para la práctica**

- Estructura general de programa para Pthreads:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
void* funcion(void *arg){
  int tid=*(int*)arg;
          printf("Hilo id:%d\n",tid);
         //Código que ejecutará cada hilo
          pthread_exit(NULL);
}

int main(int argc, char* argv[]){
 int T = atoi(argv[1]);
 pthread_t misThreads[T];
 int threads_ids[T];

for(int id=0;id<T;id++){
          threads_ids[id]=id;
          pthread_create(&misThreads[id],NULL,&funcion,(void*)&threads_ids[id]);
}

for(int id=0;id<T;id++){
          pthread_join(misThreads[id],NULL);
}
return 0;
}
```

*NOTA: La variable T recibe el número de hilos como primer parámetro de programa. Por lo tanto, si queremos ejecutar el programa para que cree 8 hilos la línea de comando será:*

```
./programa 8
```

- *Es común que los algoritmos paralelos tengan varias etapas de ejecución. NO deberían crearse Hilos una y otra vez por cada una de las etapas. Los Hilos deberán crearse una única vez y cada etapa debe estar separada por una barrera de sincronización.*
- *Compilar en Linux gcc:*

```
gcc -pthread  –o salidaEjecutable archivoFuente
```

- *Calcular el speedup y la eficiencia del algoritmo paralelo respecto del algoritmo secuencial.*

1. Paralelizar la multiplicación de matrices cuadradas de N*x*N. Obtener el tiempo de ejecución para N=512, 1024 y 2048. Ejecutar con 2 y 4 threads.

2. Paralelizar un algoritmo que cuente la cantidad de veces que un elemento X aparece dentro de un vector de N elementos enteros. Al finalizar, la cantidad de ocurrencias del elemento X debe quedar en una variable llamada *ocurrencias*. Ejecutar con 2 y 4 threads.

3. Paralelizar la búsqueda del mínimo y el máximo valor en un vector de N elementos. Ejecutar con 2 y 4 Threads.

4. Paralelizar un algoritmo que calcule el valor promedio de N elementos almacenados en un vector de tamaño N. Ejecutar con 2 y 4 Threads.

5. Paralelizar la ordenación por mezcla de un vector de N elementos. Ejecutar con 2 y 4 Threads.

6. Paralelizar un algoritmo que determine si un vector de N elementos es monotónico. Un vector es monotónico si todos sus elementos están ordenados en forma creciente o decreciente.

7. Pensar un diseño y posible implementación al problema de las N Reinas sobre memoria compartida utilizando Pthreads. ¿Cómo distribuir el trabajo cuando N es menor/mayor al número de hilos?