# Fundamentos de Sistemas de Operação

## MIEI 2018/2019

## Laboratory session 9

## Objectives

Do parallel counting using Pthreads API; the use of *mutexes*.

## Counting numbers in an array

The following code belongs to a program that counts the number of times a specified number appears in an array (vector) of integers, and measures the time it takes to complete such operation. The array is initialized with random integer numbers from 0 to 3. The `docount` function counts the number of elements equal to number given in `tofind` (e.g. 3).

```
int *array;
int count = 0;
int tofind = 3;

void docount(void) {
       for (int i=0; i < SIZE; i++) {
              if (array[i] == tofind) {
                     count++;
              }
       }
}

int main( int argc, char *argv[] ) {
       struct timeval t1,t2;

       array= (int *)malloc(SIZE*sizeof(int));
       tofind = 3;

       srand(0);
       for (int i=0; i < SIZE; i++) {
              array[i] = rand() % 4;
       }

       gettimeofday(&t1, NULL);
       docount();
       gettimeofday(&t2, NULL);

       printf("Count of %d = %d\n", tofind, count);
       printf("Elapsed time (ms) = %lf\n",
              ((t2.tv_sec - t1.tv_sec)*1000000 + (t2.tv_usec - t1.tv_usec))/1000.0 );
       return 0;
}
```

Measure the time that this sequential program takes to count number 3 in the array. After that, rewrite this program to use two (or more) threads to do that same operation, using `pthread_create`. The `docount` function is prepared to look just to a partition of the data, hence you can easily use it for the several threads, so that each thread counts the numbers in some part of the array. Don't forget to compile with "-pthread" option.

**Is the counting correct?** Measure the time for your new multithreaded version.

Produce a new, corrected, version of the program controlling the concurrency, using *mutexes* if needed. Is this version faster or slower that the previous ones?

There are several possible correct solutions, some with better performance than others. You can even design a solution that **doesn't need** *mutexes*. Try to implement more than one. Explain the execution time differences between all program versions.