Operating Systems 4. Threads

This practical activity is intended to familiarize the student with the routines of the Operating System for management of lightweight processes (threads) and the mechanisms of synchronization and protection among them. For this practical activity, we will make use of POSIX threads.

Suggested Shell commands: `man, ps, top, vmstat, perf, fg, bg, &, ...`
Suggested C funcitons: `pthread_create, pthread_self, pthread_exit, pthread_join, pthread_mutex_lock, pthread_mutex_unlock, qsort`
Include (header files): `<stdlib.h> <unistd.h> <pthread.h>`

*** Note: Find out how these commands and C functions will help you in the development of this practical activity.**

1. Lightweight Process Management in C (PThreads)

In this section, we will design and develop a program that can read a file that contains N integers and order them in ascending order. To do this we will divide the work into lightweight processes to speed up the ordering tasks. In the process, we will also be calculating the maximum and minimum values and the arithmetic mean, using blocking mechanisms that ensure the correct access to shared variables (maximum, minimum and arithmetic mean).

The student should use the following program to create the file containing the integers:

```
#include <stdio.h>
#include <stdlib.h>

void main(void)
{
        int tam=10000; //Defining the number of elements
        int i;
        FILE *fich;

        fich=fopen("myfile","w");
        fprintf(fich,"%d\n",tam);

        for(i=0;i<tam;i++)
        {
         fprintf(fich,"%d\n",rand()%1000);
        }
        fclose(fich);
}
```

To perform the sorting process, we will use the quick sort function (qsort in C) `void qsort(void *base, size_t nitems, size_t size, int (*compar) (const void*, const void*))`

Use the **`time`** command to find out the execution time as a function of the number of light processes in which we divide the resolution of the problem.

Launching in background, or on another terminal, check that the threads are created correctly using the command line.

Note: You need to compile with –pthread option to be able to use POSIX threads.

2. Creating a Makefile

Define a basic **Makefile** file to help you with the compilation tasks.
The file should facilitate changing the compilation options, and should delete all the objects and compiled files through the clean command.