Exercise 1. In Aula Global, you can the code of the following program (p1.c), as well as a file Makefile that allows to compile that program (as well as the other programs described in the rest of exercises). To compile this program, run make p1.

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

#define NUM_THREADS     2
#define ITER            10

void funcion(int *id) {
        int j;
        int s;
        double k;
        int mid = *id;  // each thread receives a number (0 or 1)

        for(j=0 ; j < ITER; j++) {
                k = (double) rand_r((unsigned int *) &s) / RAND_MAX;
                usleep((int) (k * 100000)); // sleep between 0 and 100 ms
                printf("Thread %d iteration %d \n", mid, j );
        }

        pthread_exit(NULL);

}

int main(int argc, char *argv[])
{
        int j;
        pthread_attr_t attr;
        pthread_t thid[NUM_THREADS];
        struct timeval t;

        gettimeofday(&t, NULL);
        srand(t.tv_sec);     // initialization of a random seed

        pthread_attr_init(&attr);

        for (j = 0; j < NUM_THREADS; j++)
                if (pthread_create(&thid[j], NULL, (void *) funcion, &j) == -1){
                        printf("Error creating threads\n");
                        exit(0);
                }

        for (j = 0; j < NUM_THREADS; j++)
                pthread_join(thid[j], NULL);

        exit(0);

}
```

This program creates two threads, each of which executes a series of iterations in which the thread locks a random time between 0 and 100 ms and prints its identifier and iteration number. Compile (make p1) and run the above code several times. Identify the implementation problems you observe.

Exercise 2. Modify the above program so that each of the threads prints its identifier (0 or 1) in a correct way. The name of this program will be p2.c. To compile it use make p2.

Exercise 3. Modify the previous program so that the following code for each of the threads runs under mutual exclusion. Use mutex for this. The name of this program will be p3.c. To compile it use make p3.

```
k = (double) rand_r(&s) / RAND_MAX;
usleep((int) (k * 100000)); // sleep between 0 and 100 ms
printf("Thread %d iteration %d \n", mid, j );
```

Which problems do you observe?

Exercise 4. Modify the program in Exercise 2 so that the two threads run the code:

```
k = (double) rand_r(&s) / RAND_MAX;
usleep((int) (k * 100000)); // sleep between 0 and 100 ms
printf("Thread %d iteration %d \n", mid, j );
```

alternately, first the thread 0, then the 1, then the 0 and so on. The name of this program will be p4.c. To compile it, use make p4.

Exercise 5. Modify the previous program so that 10 threads are created for executing all the code of the function `function`. All threads have to toggle the execution of the function loop: first the 0, then the 1 ..., then the 9, then the 0, then the 1, and so on. The name of this program will be p5.c.

Exercise 6. The code of a producer-consumer program (p6.c) is provided in the supporting material. The program copies a file passed as a parameter to another file. The producer process reads from the file and inserts the characters into a buffer, and the consumer extracts the characters from the buffer and writes them to the output file. Compile, run, and analyze program operation. Then modify the previous program so that the execution of the two processes is correct. Note that when the producer process finishes reading the file you must notify the consumer of that event.

Exercise 7. In the supporting material, we provide the code of a program that aims to solve the problem of the readers-writers (giving priority to the readers). Run the program and identify the existing problem. Then modify the previous program so that the execution of the processes is correct.

Note: The Makefile file provided allows you to compile the 7 programs described above.