

Barber shop

1 Description of the problem

The context of this exercise is a barber shop where there are `nbarbs` barbers and `nchairs` chairs. Each barber receives a client and serves him using a chair. Because there are fewer chairs than barbers, some mechanism must be put in place to handle conflicts. Two scenarios are defined:

1. A barber can only use one chair. Each barber has an integer identifier `barber = {0, 1, ..., nbarbs - 1}` and the same for each chair `chair = {0, 1, ..., nchairs - 1}`. Therefore, in this scenario, each barber can only use `chair = barber % nchairs` where `%` is the modulo operator.
2. A barber can use any chair.

This exercise is about parallelizing the serving of all clients so that barbers can work at the same time depending on the number of available chairs.

2 Package content

In the `barber_shop` directory you will find the following files:


- `main.c`: this file contains the main program which first calls the `init_data` routine which initializes the data. The main program then calls the `barber_shop_seq` routine that runs the barber shop sequentially (i.e., one barber only). Then it calls the `barber_shop_par_fixed` and `barber_shop_par_any` which have to be parallelized according to the two scenarios above. **Only this file has to be modified for this exercise.**
- `aux.c`, `aux.h`: these two files contain auxiliary routines and **must not be modified.**

The code can be compiled with the `make` command: just type `make` inside the `barber_shop` directory; this will generate a `main` program that can be run like this:

```
$ ./main nbarbs nchairs nclients
```

where `barbs` is the number of barbers, `nchairs` the number of chairs and `nclients` the number of clients to serve. Each client takes a random time; at the end of the day, all clients have to be served. The program prints messages if multiple barbers attempt to use the same chair at the same time, or if multiple barbers try to serve the same client. Also, note that it is not possible to receive multiple clients at the same time.

3 Assignment

-  At the beginning, the `barber_shop_par_fixed` and `barber_shop_par_any` routines are, simply, a copy of the `barber_shop_seq` one. Modify these two routines to achieve the parallelization, respectively, in the two scenarios described above. In the parallelization, each thread represents a barber. **This exercise must be done using OpenMP locks to prevent concurrent use of chairs.**

Advice

- Note that it is possible to define and use arrays of locks like this:

```
omp_lock_t *locks;  
  
locks = (omp_lock_t*)malloc(nlocks*sizeof(omp_lock_t));  
  
...
```