


# BE Systèmes Concurrents


07/12/2016

This BE is made of five different exercises of different difficulty for a total of 100 points:

- Part 1 derivative\_free:** This exercise is about parallelizing a simple derivative-free minimization algorithm. **25 points.**
- Part 2 linked\_list:** This exercise is about parallelizing a linked list traversal with two different methods. **15 points.**
- Part 3 matrix\_multiplication:** An exercise on using task dependencies to parallelize the product of two dense matrices. **15 points.**
- Part 4 reduction:** This exercise is about parallelizing a method which performs a reduction of all the elements of an array. **25 points.**
- Part 5 synchronization:** An exercise on implementing different ways to avoid data hazards in a parallel code. **20 points.**

For each exercise, detailed explanations and instructions are given in the `subject.pdf` file inside the corresponding directory.

All the exercises include some coding tasks: these tasks consist in writing, compiling and executing some OpenMP parallel code and are identified by the keyboard symbol .

Some exercises also include some analysis identified by the pencil symbol . Please write the answer to these questions in the `responses.txt` file in the topmost directory of the BE package. **Answers can be written in French.**

## Important

**Before starting the exercises**, execute the `environment.sh` script like this:

```
$ source ./environment.sh
```

this will set some environment variables needed by the exercises.

Once you have finished execute the `pack.sh` script like this:

```
$ ./pack.sh
```

This will generate a package containing the code you have developed and the responses you have provided and automatically send it to the supervisors.

**Before leaving verify with the supervisor in your room that the package has been received.**

## Some technical details

- Never use the `num_threads` clause of the `parallel` construct but, instead, choose the number of threads to use with the `OMP_NUM_THREADS` environment variable. With this method, the number of threads to be used by the program can be chosen prior to its execution by typing the command

```
export OMP_NUM_THREADS=4
```

for, example, if 4 threads are desired.

- **All the parallel codes must work for any number of threads!** Even if you are shown examples with a fixed number of threads, you cannot assume in your code that the number of threads will always be the same as in the example.
- OpenMP tasks are very powerful and convenient. However they are also very heavy and difficult to use. Therefore, unless you are explicitly asked to use tasks, always try all other options before choosing to use tasks.
- In some exercises you may need to allocate an auxiliary array. This can be done with the `malloc` function. For example, the allocation of an array of integers of size `n` can be done like this:

```
int *newarray;  
newarray = (int*)malloc(n*sizeof(int));
```