Procedures

1 Procedures

This exercise is about parallelizing a code that executes a number of procedures, each consisting of a variable number of steps that use different resources.

The code trivially consists of a main loop where, a each iteration, a procedure is fetched from a list of procedures using the `get_procedure` routine. After the procedure is fetched, a second loop is started that goes through all the steps in the procedure and, for each of them, executes the corresponding action. Each action uses a resource that can be of type printer, cpu, screen, disk or memory through the routines `use_printer`, `use_cpu`, `use_screen`, `use_disk` and `use_memory`, respectively. None of the above-mentioned routines (including `get_procedure`) is thread-safe, which means that two or more threads cannot use the same resource or get a procedure at the same time.

2 Package content

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

- `main.c`: This file contains the code that implements the code described above. This reads from command line the number of procedures that have to be executed. **Only this file must be modified for this exercise.**
- `aux.c, aux.h`: these two files contain auxiliary routines and declarations and **must not be modified**.

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

```
$ ./main np
```

where `np` is the number of procedures to execute.

3 Assignment

- ⌨ The objective of this exercise is to parallelize the code of the `main` program in order to reduce its execution time. Procedures can be executed in any order but the steps of one procedure must be executed in order, one after the other.