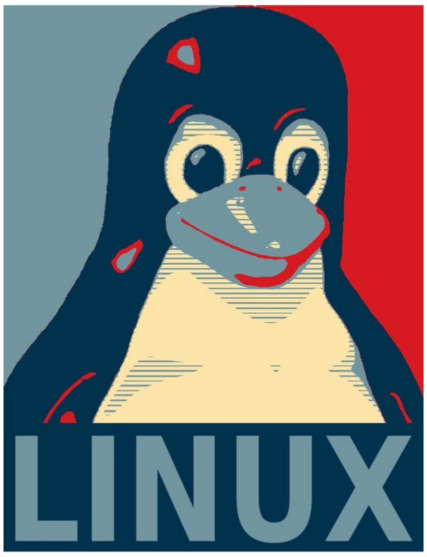
# State Challenge



# State Challenge

## *Part 1: implement and unit test the Microwave state diagram*

*Guideline: ± 0,5 days of work*

Study the StateBehaviour presentation and design a state diagram for the given microwave oven in a group of 6-10 people. A start has been made, it can be found in the Microwave/doc directory.

The product dir contains code for the given state, please study the implementation and implement at least one extra state from your state diagram in this project. The test dir contains unit tests with mocks for the given state, please unit test your implemented state well.

Make sure you really understand:

- how state diagrams work,
- how you can implement them,
- how mocks work, and
- how you can test your states using mocks

before continuing with part 2!

## *Part 2: bread bake machine*

*Guideline: ± 3,5 days of work*

Take a look at the Breadbake project. In this package you'll find the following directories:

- doc: class diagram, description of the bread bake machine and description of the provided simulator
- product: user interface, interfaces and simulator code
- test: mocks and an empty test file (testStandbyState contains all mock functions you can use)

Please study the documentation carefully. What you need to do:

- Design the state diagram for this bread baking machine. You must use nested states, this will make your diagram easier to read and maintain. Implementation will be a bit more difficult, but you'll have to learn that anyway!
- Implement your state diagram, make sure your nested states are properly implemented!

Some tips:

- in the class diagram you'll find some quick and dirty solutions to easily connect the simulator to the system. E.g. EventGenerator implementing iUserActions. These choices are not pretty but are there to make testing the machine possible for you without costing too much time.
- the function of BreadBaker::Pulse is already implemented. Pulse automatically calls a public HandleEvent method. I suggest you make the various HandleState… methods public as well. For encapsulation it would be better to make them private, however that makes unit testing your code rather annoying.

Please demonstrate:

1. Documentation (in PDF):
   1.1. your state diagram(s)
   1.2. How you tested your code and what the results are
   1.3. Which known problems your implementation has (if applicable: including errors from Klocwork that you couldn't solve)
2. Your code that follows the coding standard and where applicable shows a correct application of the RAII principles.
3. Make sure your code doesn't have dead code or unnecessary comments.
4. Make sure your code doesn't have memory problems (Valgrind + Klocwork)
5. Documentation is delivered in PDF and code is delivered without object files, executables or IDE specific files present. Only source code and Makefile(s).