Tutorial / Lab n° 2
Linux Modules

1 My first kernel module: « Hello World »

1.1 Creating a « Hello World » module

Start to create a module « Hello World » with the following characteristics. The module should:

- Display a message when loading the module (« Hello World !») and when unloading the module (« Goobye »)
- Take an integer parameter `n` which should repeat the message when loading module `n` times
- Take a string parameter `str` and to concat it to the unloading message (for example « cruel world ! »)

Create a `Makefile` allowing to compile this module. Test your module with the `insmod` command.

1.2 Loading a module

You should now try to load your module with the `modprobe` command. When you have found the right instructions to achieve this, modify the `Makefile` so that all these operations should be automatic.

Modify the system configuration so that, with the `modprobe` command, your module should not be loaded with the default value of the parameters. You should set new default values for loading the module in the right configuration file. Test that all is correct.

1.3 Add some functionalities to the « Hello World » module

Add the following functionalities to your « Hello World » module:

- Display a new message when loading the module: you should display the kernel version which the module is loaded. You would take care not to use the C macro in the source code, because, this may result as always display the kernel version number which your module was compiled with. You would rather use a function

```
See linux/utsname.h
```

- Display the time since the module is loading when unloading it. You would use the `do_gettimeofday()` function to achieve that.

2 Dependences between modules

2.1 « Hello » Module

Create a new `hello` module which:

- Displays its name when loading it
- Displays its name when unloading it
- Offers a new `print_hello()` function that would display the string "Hello" (export function symbol)

2.2 « World » Module

Create a new `world` module which:

- Depends on the « Hello » module you just created

- When loading it:
  - Displays its name
  - Uses the `print_hello()` function
  - Displays ="world !"
- When unloading:
  - Displays its name

2.3 Testing dependences

Install these two new modules and configure your system so that you can verify that when you load the `world` module, the kernel will automatically load the `hello` module.

3 If you still have got time…

3.1 A more complex module: list of processes

Write a Linux module that displays the list of processes running when the module is loaded. For each process, it suffices to show its PID and the command associated with it (16 characters).

The main difficulty here is to find a pointer to the running task. To address this problem, read the source file `kernel/sched.c` in conjunction with the list of kernel symbols (`System.map`). This should allow you to find an entry point in the process list.

If the symbol that you are getting is not exported, you should retrieve its address. Initially, this address may be added in the code module (method presented during lecture).

For those who still have got time at the end of this lab, you can try to add a parameter to your module and change the Makefile to automate the calculation of the address.