In this assignment, we shall build on what we did in assignment 2 and further extend it. You shall write a dynamically loadable module that allows us to get the current date and time when the module if read from a user-mode application.

Of course the "libc" library contains API functions that allows us to obtain the current time but we are going to implement our own version instead, that's the goal of this assignment.

When reading from the module, you may either write a user-mode program that opens and reads from your device (just as if it's a file) or you may use the following command:

>> cat /dev/lab3

it shall return:

"Hello world, the current date is month, day year. The current time is hour:minutes:seconds".

Your module should:

a) Use the `misc_register()` functionality to register itself as a character device. This requires that you create a structure (`struct file_operations`) containing the function pointers (open, release, read, etc.) and pass it to `misc_register()`.
b) Define the open(), release() and read() functions. Make sure you populate their function pointers in `struct file_operations`.
c) you may test your code by either writing your own user-mode program that reads from your device (useful for debugging) or by using:
>> cat /dev/lab3 // that's if you named your device lab3 during misc_reg()

Some Linux Notes:

- Devices (or more correctly device drivers) are organized as character, block or network devices.
- Devices have major and minor numbers. The major number specifies the device type whereas the minor number specifies an instance of that type. As such, it is common that a single device driver takes care of an entire major number.
- Most major numbers are already assigned to specific device drivers so you are not supposed to use them.
- Linux uses the concept of virtual file systems, one of those is the devfs.

- Generally, when you register your device driver (i.e. your module) with the devfs using a major and a minor number,
- You can then create a node for it in the /dev directory (using the mknod command), e.g. /dev/lab3
- You can then treat it as a file and file operations such as fopen(), fprint, fread, etc. are routed to functions that your module handles.

- Instead of using this complex registration and node creation process, we shall use the `misc_register()`, which handles both steps automatically. It registers us under device major number 10 (amongst with many other devices besides us using the same major number but different minor numbers).
  - You need to provide a minor number to `misc_register()`, the device's name ("lab3") and a pointer to the file operations.
  - In order to avoid collisions, use the `MISC_DYNAMIC_MINOR` macro as the minor number when you register your device.
- For information regarding misc_register(), see: http://www.linuxjournal.com/article/2920.
- You may find further information at kernel.org or lwn.net
- For reasons that shall be explained when we study virtual memory (towards lectures 8-9), you cannot directly use the pointers provided by the read() function in `struct_file_operstions`. Instead you need to use:

```
unsigned long copy_to_user(void __user *to, const void *from, unsigned long n);
```