System Programming – Project 1

For this project,

- you are required to write a system call which sets the value of a flag (`myFlag`) in the task descriptor of a process.
- you are required to modify the "fork" and "exit" system calls to change their behavior based on the value of the flag.

The prototype for the system call will be

```
long set_myFlag(pid_t pid, int flag);
```

The `myFlag` flag can be 0 or 1. The system call sets the value of this flag for the process with the given `pid`. Only processes having root privileges can successfully execute this system call. On error, the `set_myFlag` system call returns an appropriate error code. Otherwise, it returns `0`. (To see a list of default error codes, refer to the manual pages using "man errno".)

You are required to modify the behavior of the `fork` system call as follows:

- If the process which has `myFlag=0` makes a `fork` system call, default actions will be performed.
- If the process which has `myFlag=1` makes a `fork` system call, NO child processes will be created and an appropriate error code will be returned. (To see a list of default error codes, refer to the manual pages using "man errno".)
  Please note that this action will be performed ONLY when the "nice" value of the process with `myFlag=1` is greater than 10 (i.e. priority is greater than 30). Otherwise, the `fork` system call will work normally.

You are required to modify the behavior of the `exit` system call as follows:

- If the process which has `myFlag=0` makes an `exit` system call, default actions will be performed.
- If the process which has `myFlag=1` makes an `exit` system call, all siblings of the exiting process, i.e. all processes that have the same parent pid (ppid) as the exiting process will be terminated along with the process itself. **Please note** that this action will be performed ONLY when the "nice" value of the process with `myFlag=1` is greater than 10 (i.e. priority is greater than 30). Otherwise, the `exit` system call will work normally.

You are also REQUIRED to write test programs to fully demonstrate all features of your project during the demo.

References:

- "Understanding the Linux Kernel, 3rd Edition" by Daniel P. Bovet, Marco Cesati (Publisher: O'Reilly Pub, 2005) which is freely accessible from the ITU Library through Safari e-books.

Important notes!

- Pay attention to the general guidelines for projects.
- You are required to submit the following files through the Ninova system as a zip file:
  - Source codes of all kernel code files you modify,
  - Source codes of your test programs and information on how to use these programs,
  - A text file listing only the changed parts in the code files you modified (Hint: Use the diff command to compare files line by line).
- Make sure to return a meaningful error code when an error occurs.
- Don't blindly copy any code from other sources.
- Each member of the team must make an individual submission, even though the submitted files are the same for all members.
- Team members will be graded individually based on their performance in the lab session and the submitted team project. Students who are not present during the lab session will not receive a grade for the project, even though they may have made a submission through the Ninova system.

Any form of cheating or plagiarism will not be tolerated. The submitted work should be the work of the team; collaboration or code sharing between different teams will be regarded as cheating. Cheating also includes actions such as, but not limited to, submitting the work of others as one's own (even if in part and even with modifications) and copy/pasting from other resources, including Internet resources, (even when attributed). Serious offenses will be reported to the administration for disciplinary measures.