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 "exit" system call to change its 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 message. 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 "exit" system call as follows:

- If the process which has `myFlag=0` makes an `exit` system call, the default actions will be performed.
- If the process which has `myFlag=1` makes an "exit" system call, all children of 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.

To achieve this behavior, you need to:

1. Add a new field to the task descriptor which is defined in the `/include/linux/sched.h` file. The name and type of the field is `int myFlag`
   Note: This field should be added to the end of the task descriptor. Why is this so?
2. Modify the code used by the kernel when creating and initializing new processes. A newly created process should have its `myFlag` field initialized to `0`.
   (Note: Remember how Process 0 is created in a Linux system. The INIT_TASK macro in `/include/linux/init_task.h` initializes the process with `pid=0`. Other processes are created and initialized by the code in the `/kernel/fork.c` file.)
3. Write a system call which changes the value of the `myFlag` field in the task descriptor if the caller process has root privileges. Add your system call to the kernel.
   - Look in the `/include/linux/sched.h` file to see how you can obtain the pointer to the task descriptor of a process given its pid.
4. Modify the "exit" system call which can be found in `/kernel/exit.c` to achieve the above described actions.
   - "`current`" points to the task descriptor of the current process. Include `linux/sched.h` file to use it.
   - `list_for_each` and `list_entry` macros may be useful for iterating over child processes.

5. Write test programs to demonstrate your project.
   - Write a short test program that accepts the pid of the process and the flag value as input and makes the `set_myFlag` system call. The test program should output the return value of the system call. Experiment by running the program with and without root privileges.
   - Write a short test program to demonstrate how the modified `exit` system call works. The test program should output the return value of the system call.
     - Experiment by running the program with the two flag values (0 or 1).
     - Experiment by running the program on processes with different priorities (nice values).

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 offences will be reported to the administration for disciplinary measures.