System Programming – Project 2

For this project, you are required to write a system call which sets a flag (`myFlag`) in the task descriptor of a process. When this flag is set to `1`,

- the process is not listed in the `/proc` file system and cannot be seen using "`ps`" or "`top`",
- it cannot fork any new processes,
- when it makes the "exit" system call, all children of the exiting process will be terminated along with the process itself.

The prototype for the system call will be

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

The `myFlag` flag can be `0` (for OFF) or `1` (for ON). The system call changes the value of this flag. 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".)

Please note that the following actions 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, both the "fork" and "exit" system calls will work normally (even if `myFlag` may be set to 1).

- If the process which has `myFlag=1` makes a `fork` system call, no new processes will be created and the `fork` system call will return an appropriate error message. (To see a list of default error codes, refer to the manual pages using "man errno".)
- 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.

To achieve this, you need to:

1. Add a new field to the task descriptor. The name and type of the field is:
   ```
   int myFlag;
   ```
   Note: This field should be added to the end of the task descriptor. (Note: Why?)
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: Learn how the process with `pid=0` is created and initialized in Linux.)
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.
4. Modify the code that generates the `/proc` filesystem so that if the `myFlag` field of a process is set to `1,` the process is not included.
5. Modify the "fork" and "exit" system calls to achieve the above described actions.
6. 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.
7. Write a short test program that makes a `fork` system call. The test program should output the return value of the `fork` system call. Experiment by running the program with the two flag values. Experiment by running the program on processes with different priorities.

8. Write a short test program to demonstrate how the modified `exit` system call works. Experiment by running the program with the two flag values. Experiment by running the program on processes with different priorities.

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.

Please read the following carefully!

- 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,
  - 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).
- Each member of the group must make a submission, even though the submitted files may be the same for all group members.
- Group members will be graded individually based on their performance in the lab session and the submitted group 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 product of the group itself; collaboration or code sharing between different groups will be accepted as plagiarism. This 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 (even when attributed). Serious offences will be reported to the administration for disciplinary measures.