System Programming – Project 2

For this project, you are required to do the following:

Part 1: Write a system call which either only terminates all the children of a given process or terminates all children, all siblings and their children of a given process. (Note: The process itself is not terminated). Only processes having root privileges can successfully execute this system call.

The prototype for the system call will be

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

`pid` is the process id of the process
`flag` determines the behavior of the system call:

0: only the children of the process with given `pid` is terminated.
1: the following processes are terminated:
- children of the process with given `pid`
- siblings of the process with given `pid`
- children of all the terminated sibling processes

Part 2: The "exit" system call will also be modified. If the "nice" value of the process which has executed the "exit" system call is greater than 10 (i.e. priority is greater than 30), then the "exit" system call will also use the new system call (`my_process_terminator`) you have written with `flag=0`, i.e. all children of the exiting process will be terminated along with the process itself.
(Note: Remember what normally happens in Linux to children processes when the parent process terminates!)

To achieve this, you need to:

1. Modify the code used by the kernel when a process makes the "exit" system call.
2. Write the required system call and add it to the kernel.
3. Write test programs to test both behaviours:
   - the test program tests the "`my_process_terminator`" system call for both settings of the `flag`.
   - the test program sets its nice value to a value greater than 10 and uses the "exit" system call.

   The test programs should output the return value of the system call. Experiment by running the programs with and without root privileges.

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.

Hints:

- To see a list of default error codes, refer to the manual pages using "man errno".
- Sibling processes are processes which have the same parent process.