System Programming – Project 2

Aim:

1. Learn how to add a new system call to the Linux kernel.
2. Remember process creation and process management in Linux.
3. Understand and modify the task descriptor structure in the kernel.
4. Understand and work with the /proc file system in Linux.
5. Learn how to compile a Linux kernel.

Implementation:

For this project, you are required to write a system call which makes a process invisible, i.e. the process is not listed in the `/proc` file system and thus cannot be seen using "`ps`" or "`top`". The prototype for your system call will be

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

The `flag` can be `0` (for OFF) or `1` (for ON). The system call turns the invisibility of a process (given by its pid) OFF or ON based on the value of the flag. Only processes with root privileges can successfully execute this system call. The call returns `0` on success and `-1` on failure.

To achieve this, you need to:

1. Add a new field to the task descriptor. The name and type of the field is:
   ```
   int invisible;
   ```
   (Note: This field should be added to the end of the task descriptor.)
   If `invisible=0` process is visible
   If `invisible=1` process is invisible
2. Modify the code used by the kernel when creating and initializing new processes. (Note: A newly created process should have its `invisible` field initialized to `0`.)
3. Write a system call which sets the invisibility field in the task descriptor if the caller process has root privileges.
4. Add your system call to the kernel.
5. Modify the code that generates the `/proc` filesystem so that if the `invisible` field of a process is set, it is not included.
6. Recompile the kernel.
7. Write a short test program that accepts the pid of the process and the invisibility status information as input and makes the `set_invisibility` system call. The test program should output the return value of the system call. Experiment by running the program with and without root privileges.

References:

- Please read Chapters 3, 7 and 10 of the book "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:

1. The task (process) descriptor structure is defined in the `/include/linux/sched.h` file.
2. Remember the process hierarchy and the process creation steps in Linux.
   - The INIT_TASK macro in `/include/linux/init_task.h` file initializes the process with pid=0.
   - Processes are created and initialized by the code in the `/kernel/fork.c` file.
3. Remember that the task list and the task descriptors may be modified by more than one process at a time. Therefore any operation performed on them should be either atomic operations or should be protected by setting locks. You may look in the `/kernel/sched.c` file to see examples of how the task list and the task structures are accessed using the appropriate locking mechanisms.
4. Look in the `/kernel/sched.c` file to see how you can obtain the pointer to the task descriptor of a process given by a pid.
5. "`current`" points to the task descriptor of the current process and the corresponding macro is found in the `include/linux/kernel.h` file.
6. `fs/proc/array.c: proc_pid_stat()` defines the format and fields to write in the `/proc/pid/stat` file.
7. `fs/proc/base.c: proc_pid_readdir()` is used to read the `/proc/pid` directories.

Note: This assignment is a part of:
`http://elearning.algonquincollege.com/coursemat/ayalac/cst8263/Assignments/03_invisible.pdf`