Exercise 1

Develop an access control system that prohibits an executable from invoking the “fork” system call using two different approaches: (i) the LD_PRELOAD and (ii) the ptrace system call.

**Implement the access control system using a library with LD_PRELOAD.**
LD_PRELOAD gives you the ability to instruct the linker to bind symbols provided by a shared library before other libraries. You are required to override the “fork” system call with your own version in order to provide the functionality this assignment asks.

Note:
- Create a folder named `stop_fork_ldpreload` and store your source code there.
- Develop a simple testing application (test.c) to demonstrate the task.
- For this exercise, the files that you are required to submit are the following: (i) inject.c, (ii) test.c, (iii) Makefile.

**Implement the access control system using the ptrace system call.**
The ptrace() system call provides a means by which one process (the "tracer") may observe and control the execution of another process (the "tracee"), and examine and change the tracee's memory and registers\(^1\). For this assignment, you are required to use ptrace to prohibit an executable from invoking the “fork” system call.

Note:
- Create a folder named `stop_fork_ptrace` and store your source code there.

---
\(^1\) [http://man7.org/linux/man-pages/man2/ptrace.2.html](http://man7.org/linux/man-pages/man2/ptrace.2.html)
Exercise 2

Generalize your previous solution as follows: The access control system will take as input a file of all system calls that should be prevented from being invoked. At the end of your execution, you need to output some statistics: (i) the number of system calls that are prohibited to execute, (ii) the number of unique system calls that your access control system successfully invoked, (iii) the total number of system calls that the executable generated.

Note:
- Create a folder named blacklist_ptrace and store your source code there.
- Develop at least a simple testing application (e.g., test.c) to demonstrate the task.
- For this exercise, the files that you are required to submit are the following: (i) inject.c, (ii) test.c, (iii) blacklist.txt, (iv) Makefile.

Exercise 3

Ten friends decided to put their life savings in a bank account. The account is completely anonymous and can be accessed only with a password. The password is an integer number. They decided to let the money stay in the bank and withdraw it on January 2nd 2050. To make sure that no one withdraws the money earlier they decided to split the password into 10 pieces. However it was suggested that if one of them is not interested in the money, or worse, does not want the rest to have their share of the money, he may simply not appear on January 2nd 2050.

For this reason they decided that if nine (any nine) of them are present they should be able to withdraw the money. However, if eight or less are present they should not be able to withdraw the money.

To share the password they developed the following scheme:
I. They developed an 8th degree polynomial: \( f(x) = a_8 x^8 + a_7 x^7 + \ldots + a_1 x + a_0 \) where \( a_0 \) is the password and \( a_i \) are numbers chosen randomly.
II. They decided to give one point of the polynomial to each friend: the first friend would take \( f(1) \) the second \( f(2) \), the third \( f(3) \), and the last one would take \( f(10) \).

Write a program to share the password and a program reconstruct the password.

Your program will have two functionalities:
1. “a.out split password” will create 10 points
2. “a.out join share_1 share_2 \ldots share_9” will recreate the secret (password), where \( \text{share}_i = (i, f(i)) \) a tuple containing one point of the polynomial
Note:
- Create a folder named `secret_sharing` and store your source code there.
- For this exercise, the files that you are required to submit are the following: (i) `split.c`, (ii) `join.c`, (iii) `Makefile`.

Submission instructions:
- You need to create a `Makefile` to compile your programs.
- You need to create a `Readme` file with your name, your login and a short description of your implementations.
- You should submit a folder named `assign2_yourAM` with your source code (`stop_fork_ldpreload`, `stop_fork_ptrace`, `blacklist_ptrace`, `secret_sharing`) until the deadline.