Read Tanenbaum, Bos: Chapter 2.2, 9.6 "Authentication" (=9.4 in the 3rd Ed), 10.3, 11.4

Exercises

1. In a typical multithreaded process:
   (a) Does each thread have its own memory address space?
   (b) Does each thread have its own stack?
   (c) Does each thread have its own set of registers?
   (d) Does each thread have its own file descriptor table?

2. Our Linux server, turing.slu.edu, has 8 CPU cores. Suppose a difficult computation takes 24 hours to run on turing. The program is single threaded, but can be effectively multithreaded.
   (a) How fast would we hope it can run using 4 threads?
   (b) How fast would we hope it can run using 8 threads?
   (c) How fast would we hope it can run using 16 threads?

3. Give an example of a program that would benefit from multiple threads even on a single processor machine.

4. POSIX `pthreads`: are they kernel threads or are they user threads?

5. Suppose a process has two threads, and one of the threads is waiting for input (maybe it called `fgets`). Now the other thread calls `fork()`, so both processes have a thread waiting for input. What should happen when the user finishes typing the input?

   Bonus: What does happen?

6. Suppose two users have the same password. Will their entries in the password file be the same?

7. Suppose you have access to the password file on a machine, and passwords are salted and hashed. Does the salt make it harder for you to guess the administrator's password?

8. (a) How many 6 character passwords are there that use only numbers and lowercase letters?
   (b) Suppose a cracker builds a lookup table by computing the hash for each of these passwords and storing them in a file. How large is the table if each hash is 12 bytes?
   (c) Now suppose 14 bits of salt are added to each password before hashing. How large is the table for these salted passwords?