Assignment 5

a. 1. Which of the calls above are blocking and which are not? Explain what that means? Is this a form of direct communications or indirect communications?

   connect(), accept() and read() are blocking calls and the others are not. These two blocking methods means that the current thread/task will be blocked until the method is finished.
   This is a form of indirect communications.

   2. How would you change your program to communicate between processes in a different machine?

   When communicating with a different machine, I will use “struct sockaddr_in” instead of “struct sockaddr_un”. There are two members called “in_port_t sin_port” and “struct in_addr sin_addr” so that I can assign the IP address and PORT number.

b. 1. Do you expect a difference when running with a single CPU core versus multiple cores (assuming each process is executed on a separate CPU)? Explain your answer. Hint: You may want to search for the term "sequential consistency" or "loose memory models".

   Yes. When running with multiple cores, the final value will not be 0. Because on x86 CPUs, the compiler may reorder the statements of read-after-write to different locations to achieve better performance so that the writing operation will occur before reading. And when running with a single CPU, if n is bigger than 1e6, the program will run very slowly. Because under the Peterson’s method, after every time one thread modifies counter, it will go into the while loop to wait for the other thread, in each time slice, the counter will be only modified once. And when n is large, the program will be very slow. But when n is small, one thread will finish before the other is started.

   2. If you are developing an operating system, how would you implement the synchronization primitives then?

   When implementing the synchronization primitives, I will add memory barrier to avoid memory reorder.