Problem 1. General Questions

1. Because threads are designed to cooperate to have better performance under a process, they will call `thread_yield` to give up their own memory for other thread when scheduling are prioritizing all the threads for the benefit of the whole operating system.
2. It's much faster since every detail about implementation doesn't rely on kernel, it only utilize pthread library to support synchronizing, scheduling threads, and it doesn't have to move from user mode to kernel mode back and force. However, since there's no chance of rewriting the whole operating system, if one thread uses system call for blocking, every threads in that process will be blocked.
3. No, fork is designed for single-threaded process, for a multithread process, only the thread which called fork can have a duplicate in its child process, while other thread in that process will be ignored. Therefore, when a single-threaded process is waiting for the input from the keyboard, only one thread which is in ready state will utilize system call to fork and get input, so there will be no issue.
4. Those system calls like function should be referred to `ntdll.dll` in windows, which defines all the Win32 API functions, a solution is to translate those system calls to your own functions relying on the api, however, it won't be as efficient as original unix system calls.

Problem 2. C programming

Please refer README.md

Problem 3. Research on POSIX

The Portable Operating System Interface (POSIX) is a set of standards released by IEEE Computer Society to maintain the compatibility among operating systems. It has formally defined application programming interface (API), command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.

The reason why POSIX exists is that for Unix-like OSs, there are a majority of distribution that are not compatible with each other on the market, and therefore it requires a common specification to cover all these standards.

POSIX standarizes user command line scripting, user-level programs, services and utilities (including awk, echo, ed), along with required program-level services (including basic I/O: file, terminal, and network among all these distributions. POSIX also defines a standard threading library API which is supported by most modern operating systems.