Ex.1 -- General questions

1. Why should a thread voluntarily release the CPU?

There's no clock interrupt to actually enforce multiprogramming as there's with processes. Thus it's important for threads to be polite and voluntarily surrender the CPU from time to time to give other threads a chance to run.

2. What's the biggest advantage/disadvantage of user space threads?

Advantage: The entire user space thread switch occurs in user space, and such a thread switch is at least an order of magnitude faster than falling into the kernel. In addition, a user-level threads package can be implemented on an operating system that does not support threads.

Disadvantage: If a thread is blocked, all of the other threads will be blocked. And similarly, if a thread is running, it will not be stopped unless it gives up CPU voluntarily.

3. If a multithreaded process forks, a problem occurs if the child gets copies of all the parent’s threads. Suppose that one of the original threads was waiting for keyboard input. Now two threads are waiting for keyboard input, one in each process. Does this problem ever occur in single-threaded processes?

For a single-threaded process, it will be blocked when waiting for keyboard input. Therefore, the process will not be able to fork.

4. Many UNIX system calls have no Win32 API equivalents. For each such call, what are the consequences when porting a program from a UNIX system to a Windows system?

It may not work properly.

Ex. 2 -- C programming

README: `readme.md`

related files: `src.zip`

Ex. 3 -- Research on POSIX

The **Portable Operating System Interface (POSIX)** is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines both the system- and user-level application programming interfaces (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.

In the Unix world, the most popular application programming interfaces are based on the POSIX standard. From a purely technical point of view,POSIX is a set of IEEE standards whose goal is to provide a portable operating system standard that is broadly based on Unix. Linux is compatible with POSIX POSIX is a great example of the relationship between apis and system calls. On most Unix systems, there is a direct relationship between API functions defined according to POSIX and system calls. In fact, the POSIX standard was modeled after the interface of early Unix systems On the other hand, many operating systems, like Windows NT, despite having nothing to do with Unix, also provide POSIX-compatible libraries. Linux system calls, like most Unix systems, are provided as part of the C library, as shown in Figure 5-1 As shown in Figure 5-1, the C library implements the major Unix apis, including the standard C library functions and system calls. All C programs can use the C library. Due to the characteristics of THE C language, other languages can easily use the C library