VE482
Introduction to Operating Systems

HOMEORK 3

October 20, 2021

Yihua Liu 518021910998

Ex. 1 — General questions

- Why would a thread voluntarily release the CPU?
  Threads share the same address space and can access the same data. Programmers can use `thread_yield()` to release the CPU when a thread is no longer needed in order to save resources for the application.

- What is the biggest advantage/disadvantage of user space threads?
  The biggest advantage of user space threads is that they can be scheduled independently from kernel traps so that less context switch would be done, which improves the performance. The biggest disadvantage of user space threads is that if one thread fails, then the whole process would be interrupted.

- 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?
  No, it does not, because for single-threaded processes, if they are waiting for keyboard input, they cannot be forked.

- 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?
  Some UNIX system calls will be replaced by platform-agnostic APIs, so the performance of the programs will be declined.

Ex. 2 — C programming

The goal of this exercise is to improve the programming skills and get more familiar with pointers and function pointers.

- Implement a linked list structure containing two pointers of type `char` and `void`. It should be possible to at least add elements to the list.

- Knowing that the `void` pointer in the structure could contain some `char*`, `int`, or `double`, write a search function for this linked list.
- The linked list will store elements read from an ASCII file where each line is in the format `somestring=somedata`. The type of the data is defined in the filename; for instance a file containing unsorted integers will be named `rand_int.txt`. Implement the necessary functions to read and write such files.
- Use a function pointers to compare and sort the elements from the structure with respect to the data field. Implement the following sorting orders: increasing, decreasing, and random. The filename is `sortingtype_dataype.txt`, where `sortingtype` is `rand`, `inc`, or `dec`.
- Write a function to test the implementation.

See attached *.c and *.h files.

Ex. 3 — Research on POSIX

Write a few paragraphs about the POSIX standards. What are they, why do they exist, what kind of things are included in the norms.

POSIX standards, the Portable Operating System Interface standards are defined by IEEE. More specifically, they are designated as IEEE 1003 and ISO/IEC 9945 currently.

They exist because IEEE wants to maintain the compatibility between operating systems. It basically starts from Unix, or more specifically, Unix System V.

They include system-level and user-level APIs (application programming interfaces), command line shells, and utility interfaces (such as `awk`, `echo`, `ed`, etc.). For program level services, they include basic I/O: file, terminal, and network.

According to POSIX documentation, POSIX has "System Interfaces, and Commands and Utilities (which include POSIX.1, extensions for POSIX.1, Real-time Services, Threads Interface, Real-time Extensions, Security Interface, Network File Access and Network Process-to-Process Communications, User Portability Extensions, Corrections and Extensions, Protection and Control Utilities and Batch System Utilities" and " A test suite for POSIX accompanies the standard: VSX-PCTS or the VSX POSIX Conformance Test Suite".