Total: 2 pts, 0.5 pts each

1. Given that you can create multiple threads to perform different tasks within a program, explain why you might still need to use fork(). (give *2 reasons*).

2. Which of the following is true about Thread v.s. Process (multi-answer)

a. All threads within a process share the same heap.

b. The signal dispositions are per-thread.

c. Each thread within a process can use exec() to execute a new program without affecting other threads in the same process.

d. A thread-safe function is always a reentrant function.

e. It is impossible for multiple processes to share the same mutex.

3. Which of the following is true about the process address space (multi-answer):

a. In modern Unix systems, a user program usually accesses memory via virtual addresses.

b. In modern Unix systems, processes can no longer share memory since copy-on-write is employed during fork().

c. A process can perform memory-mapped I/O using mmap(). To ensure the copied file contents are carried out to the file, one should always specify MAP_PRIVATE in the flags passed to mmap().

d. When the argument ***addr*** passed to mmap() is NULL, the OS kernel chooses the address in the kernel space at which to create the mapping.

e. Assume that a peer thread in a process uses mmap() to create an anonymous mapping, it is possible for other threads in the same process to access the mapping.

4. In Linux, you can observe the address space layout of a process via the file /proc/$PID/maps in “procfs” (i.e., /proc/). For instance, you can observe the process layout with PID 100 by “cat /proc/100/maps”.

Write a C program that does nothing but create a new anonymous memory region with addr=NULL and size=1024 via ***mmap()***. Discuss what you observe from */proc/PID/maps* when you execute the program (PID == process ID of the program).