1. OS is a sleeping beauty. It can be woken up by four kinds of events. What are these events? Give an example for each kind of events.

- Systems calls: It is to execute a system call (syscall) process placed in the argument in user stack and places syscall type in a dedicated register. After this the syscall machine instruction is executed. The OS kernel syscall interrupt handler, place results in dedicated register.
- Device Interrupts: Used in timer and I/O devices.
- Exceptions: Processes like missteps (division by 0), privileged instruction, etc.
- Context Switches: Switching from one process to another.

2. What is a process? More specifically, what does a process contain?

- A process is an executable running on an abstraction of a computer. A process is a combination of mutable data, registers, and files. A process runs on the CPU, each process contains its own registers, Memory, I/O resources, thread control, etc.

3. What is a thread? What does a thread contain?

- A thread is an abstraction of a CPU core. Each thread has its own thread ID, program counter, stack pointer set of registers and stack for local variables and return addresses. Multiple threads share address space, code instruction, most data (heap), open file descriptors, current working directory, user, and group id. Each individual thread contains.

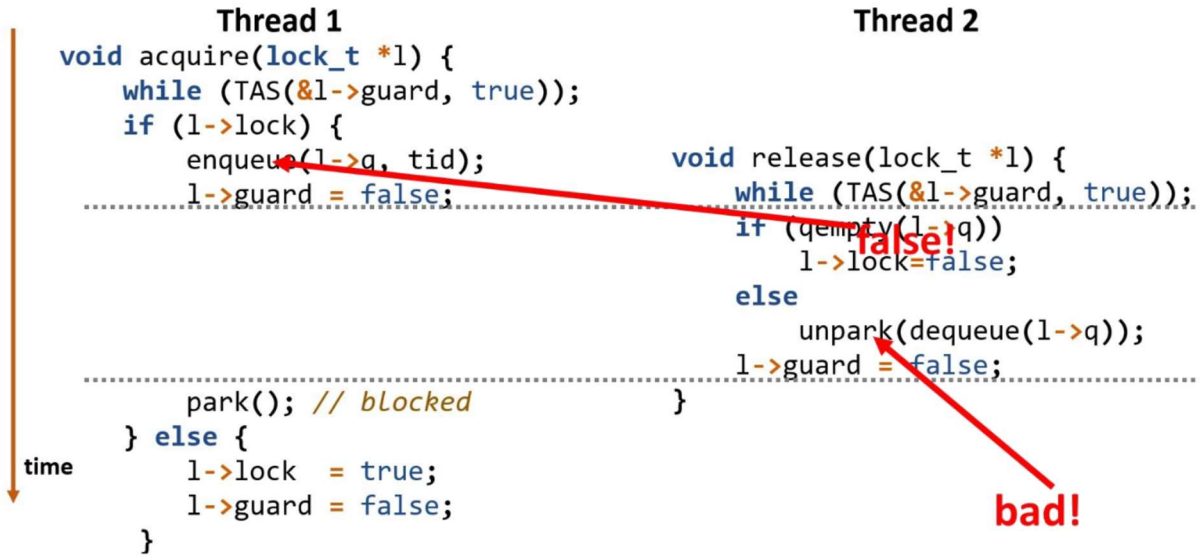
4. In 02/03 lecture slide 29, in release (), why not set *lock = false* when *unpark ()*?

```
void release(lock_t *l) {
    while (TAS(&l->guard, true));
    if (qempty(l->q))
        l->lock=false;
    else
        unpark(dequeue(l->q));
    l->guard = false;
}
```

)?

- The park () and unpark () are functions which are used for different cases. Park () function removes the thread ID from run queue and unpark () returns the thread ID to run the queue. The lock is acquired when the dequeue thread is the unpark () function, which in turn is operating in the critical section.

5. In 02/03 lecture slide 31, why does the code fix the race condition described in the previous slide?



- Thread 2 is going into critical section without acquiring the lock which leads to the incorrect output. The OS receives the plan of park () with setpark (). Which means that until thread 1 is park (), thread 2 will not be released.