1. a) No. User function executes instruction in user mode, while system call executes trap instruction to move a process into kernel mode so previleged operation can be executed

   b) No. If the both threads are reading shared variable, then concurrency error will not occur, and further more, concurrency error will not occur if only one of two threads are guarrenteed of updating the shared variable, and the other thread is reading the shared variable without importance of which ever state, before the update or after the update, it reads.

   c) No. Multiple threads under the same process share same code and data in heap memory. data in stack memory is not shared.

   d) No. The process of switching two processes process via `switch` system call occurs in kernel mode.

   e) No. External fragmentation occurs when there are holes in disk because the space cannot be filled by data blocks of a file

   f) No. Solid state drive focusing on putting close blocks together would result in a condition called wear out, and we want to avoid wear out as much as possible.

2. a) When CPU has resources to execute the thread in ready state

   b) A signal (specifically `pthread_cond_signal`)

   c) Switching of mode flag from user mode to kernel mode, saving of user registers to kernel stack

Notes

- I need to review thread