Threads and Page Faults

**Threads:**

What is the difference between a thread and a process?
xk doesn't have a thread abstraction - if you were to implement threads in xk, what fields would they have? With threads, how would the fields of the proc struct change?

What are the differences between kernel and user threads?

Why use user threads on a single cpu machine?

List some pros and cons of a many-to-one mapping of user threads to kernel threads compared to a one-to-one mapping.
Threads and Page Faults

Page Faults:

A trap 14 defines a page fault, this means that the memory address was a not a valid page for the client to manipulate.

Can the kernel cause a page fault? If so, how?

For a user process, how will you know if the page fault was caused by attempting to access the stack region of its virtual address space?
Hint: trap.c has a variable \texttt{addr} which is the address the user process tried to access.

The trapframe error code can be read with \texttt{myproc()->tf->err}.
What will the error code be if the page fault was from touching the stack region of memory?

Can the kernel cause a page fault that was meant for stack growth?

What do the fields of a page (\texttt{struct vp}) need to be after a copy-on-write fork?
Can the kernel cause a copy-on-write page fault?
What will the error code be if the page fault was from touching a copy-on-write page?
When is copy-on-write less efficient than a deep copy fork?