Question: 1. What is the purpose of copyin()?

The purpose of *copyin()* is to copy data from a user-space address to a kernel-space address.

Question: 3. In runprogram(), why is it important to call vfs_close() before going to user mode?

Note that after switching to user mode, execution never goes back to runprogram() again. Therefore if we do not call vfs_close() on a previous vfs_open before going to user mode then the resources associated with the opened file will never be freed.

Question: 4. Which kernel function is used to make a thread switch to executing user-level code?

The kernel function is *enter_new_process()* which is used in runprogram.c

Question: 5. What (briefly) is the purpose of userptr_t?

userptr_t is a pointer to a one-byte struct in userspace, for the purpose of not mixing it with pointers in the kernel space.

Question: 6. Why do you "probably want to change" the implementation of kill_ curthread()?

The default implementation does not know how to handle the fatal fault – it just invokes a panic action and shuts down the kernel (it does not clean up any of the resources used by the thread as well). We need to properly implement how this function is to handle a fault.

Question: 7. At the time that syscall() is invoked, are interrupts enabled or disabled? How about when kill_curthread() is invoked?

Interrupts are enabled both when *syscall()* or *kill_curthread()* is invoked.

Question: 8. What is the difference between copyin() and copyinstr()?
Whereas *copyin()* can copy bytes of any type of data (from user-level address to kernel address), *copyinstr()* only copies bytes of a null-terminated string (from user-level address to kernel address).

Question: 9. Which kernel function is used to open a file or device and obtain a vnode?

The function is *vfs_open(..)*
Question: 10. What operations can you do on a vnode? If two different processes open the same file, do we need to create two vnodes?

There is a big list of operations you can do on a vnode; these can be found in the *kern/include/vnode.h* file. Some of these operations are open, close, reclaim, read, write, gettype, truncate, and namefile.

If two different processes open the same file, we do not need to create two vnodes but simply increment a counter associated with the file (i.e. vn_opencount and/or vn_refcount).