Project Group 3: Code Answers

Question 1: What is the purpose of **copyout()**?

copyout() copies a block of memory from kernel space to user space.
This enables user programs to communicate with the kernel through the use of syscalls.

Question 2: In **runprogram()**, why is it important to call **vfs_close()** before going to user mode?

Calling vfs_open() allows a vnode (an abstract structure for an on-disk file) to be open in user mode. Until the vnode associated with progname is closed, runprogram() will continue to keep vnode open in memory beyond supervisor mode unless vfs_close() is called.

Question 3: Which kernel function is used to make a thread switch to executing user-level code?

enter_new_process() from kern/include/syscall.h

Question 4: What (briefly) is the purpose of **userptr_t**?

Stores the address of where the userspace memory begins.

Question 5: Why do you "probably want to change" the implementation of **kill_curthread()**?

The current kill_curthread() implementation avoids any actual trap handling and panics, thus halting the process abruptly without handling the exception.

Question 6: At the time that **syscall()** is invoked, are interrupts enabled or disabled? How about when **kill_curthread()**is invoked?

At the time syscall() is invoked, interrupts are disabled as the function simply “delegates the work of the system call to the kernel function that implements it.”
At the time kill_curthread() is invoked, interrupts are enabled because the function still needs to extract the exception code information from the register fields.

Question 7: What is the difference between **copyin()** and **copyinstr()**?

copyin() copies a block of memory of specified size from userspace to kernel space. On the other hand, copyinstr() copies a string of specified size from userspace to kernel space.

Question 8: Which kernel function is used to open a file or device and obtain a vnode?

vfs_open()

Question 9: What operations can you do on a vnode? If two different processes open the same file, do we need to create two vnodes?

vop_open, vop_close, vop_reclaim, vop_read, vop_readlink, vop_getdirentery, vop_write, vop_ioctl, vop_stat, vop_gettype, vop_tryseek, vop_fsync, vop_mmap, vop_truncate, vop_namefile, vop_creat, vop_symlink, vop_mkdir, vop_link, vop_remove, vop_rmdir, vop_rename, vop_lookup, vop_lookparent.

No, since processes point to vnodes, two processes can point to the same vnode.