CSE3211: Operating System Assignment 0

Saif Mahmud
2015-116-815 & SH-54

Tauhid Tanjim
2015-716-819 & SH-58

August 13, 2018

Questions & Answers

Q1. What is the vm system called that is configured for assignment 0?
Answer: dumbvm
Location: kern/arch/mips/conf/conf.arch

Q2. Which register number is used for the stack pointer (sp) in OS/161?
Answer: $29
Location: kern/arch/mips/include/kern/regdefs.h

Q3. What bus/busses does OS/161 support?
Answer: LAMEbus
Location: kern/arch/sys161/include/bus.h

Q4. What is the difference between splhigh and spl0?
Answer: splhigh() sets IPL to the highest value, disabling all interrupts. spl0() sets IPL to 0, enabling all interrupts.
Location: kern/include/spl.h

Q5. Why do we use typedefs like u_int32_t instead of simply saying "int"?
Answer: Since we're in a 32-bit platform, size_t, ssize_t, and ptrdiff_t can correctly be either (unsigned) int or (unsigned) long. However, if we don't define it to the same one gcc is using, gcc will get upset. So, in order to make sure that we really get a 32-bit unsigned integer as unsigned int depends on the platform, we use u_int32_t.
Location: kern/arch/include/mips/kern/types.h

Q6. What must be the first thing in the process control block?
Answer: Not Found
Location: N/A

Q7. What does splx return?
Answer: Returns old spl level
Location: kern/thread/spl.c

Q8. What is the highest interrupt level?
Answer: There are two interrupt levels, 0 and 1. 1 is defined as IPL_HIGH.
Location: kern/include/spl.h

Q9. What function is called when user-level code generates a fatal fault?
Answer: Function called when user-level code hits a fatal fault is static void kill_curthread(vaddr_t epc, unsigned code, vaddr_t vaddr).
Location: kern/arch/mips/locore/trap.c

Q10. How frequently are hardclock interrupts generated?
Answer: Hardclocks per second is 100Hz (#define HZ 100)
Location: kern/include/clock.h

Q11. What functions comprise the standard interface to a VFS device?
Answer: devop_eachopen - called on each open call to allow denying the open.
devop_io - for both reads and writes (the uio indicates the direction).
devop_ioctl - miscellaneous control operations.
Location: kern/include/device.h

Q12. How many characters are allowed in a volume name?
Answer: 32 (#define SFS_VOLNAME_SIZE 32 /* max length of volume name */)
Location: kern/include/kern/sfs.h

Q13. How many direct blocks does an SFS file have?
Answer: 15 (#define SFS_NDIRECT 15 /* # of direct blocks in inode */)
Location: kern/include/kern/sfs.h

Q14. What is the standard interface to a file system (i.e., what functions must you implement to implement a new file system)?
**Answer: fsop_sync - Flush all dirty buffers to disk.
fsop_getvolname - Return volume name of filesystem.
fsop_getroot - Return root vnode of filesystem.
fsop_unmount - Attempt unmount of filesystem.**
Location: kern/include/fs.h

Q15. What function puts a thread to sleep?
Answer: void wchan_sleep(struct wchan *wc, struct spinlock *lk);
Location: kern/thread/thread.c

Q16. How large are OS/161 pids?
Answer: 32 bit (typedef __i32 __pid_t; /* Process ID */)
Location: kern/include/kern/types.h

Q17. What operations can you do on a vnode?
**Answer: vop_eachopen, vop_reclaim, vop_read, vop_readlink
vop_eachopen, vop_reclaim, vop_read, vop_readlink, vop_getdirentry, vop_write, vop_ioctl, vop_stat, vop_gettype, vop_isseekable, 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.**
Location: kern/include/vnode.h

Q18. What is the maximum path length in OS/161?
Answer: 1024 (/* Longest full path name */ #define __PATH_MAX 1024)
Location: kern/include/kern/limits.h

Q19. What is the system call number for a reboot?
Answer: 119 (#define SYS_reboot 119)
Location: kern/include/kern/syscall.h

Q20. Where is STDIN_FILENO defined?
Answer: #define STDIN_FILENO 0 /* Standard input */
Location: kern/include/kern/unistd.h

Q21. What does kmain() do?
Answer: Kernel main : Boot up, then fork the menu thread; wait

for a reboot request, and then shut down.
Location: kern/main/main.c

Q22. Is it OK to initialise the thread system before the scheduler? Why (not)?
Answer: Yes. Because the scheduler bootstrap creates the run queue and the thread bootstrap initializes the first thread.
Location: kern/thread/thread.c

Q23. What is a zombie?
Answer: Zombies are threads that have exited but still need to have thread_destroy called on them.
Location: kern/thread/thread.c

Q24. How large is the in initial run queue?
Answer: Though we did not find any initialization of the run queue, it seems that this line of code is an indication of run queue size (curcpu->c_runqueue.tl_count = 0;)
Location: kern/thread/thread.c

Q25. What does a device name in OS/161 look like?
Answer: The name of a device is always just "device:". The VFS layer puts in the device name for us, so we don't need to do anything further. (e.g. "lhd0")
Location: kern/vfs/vfslist.c

Q26. What does a raw device name in OS/161 look like?
Answer: Name of raw device (e.g., "lhd0raw"). Is non-NULL if and only if this device can have a filesystem mounted on it.
Location: kern/vfs/vfslist.c

Q27. What lock protects the vnode reference count?
Answer: vn_countlock
Location: kern/vfs/vnode.c

Q28. What device types are currently supported?
Answer: Block Type and Character Type Devices.
Location: kern/vfs/device.c