**Question 1:**

* System/161 Version: System/161 release 2.0.8

OS/161 Version: base system version 2.0.3

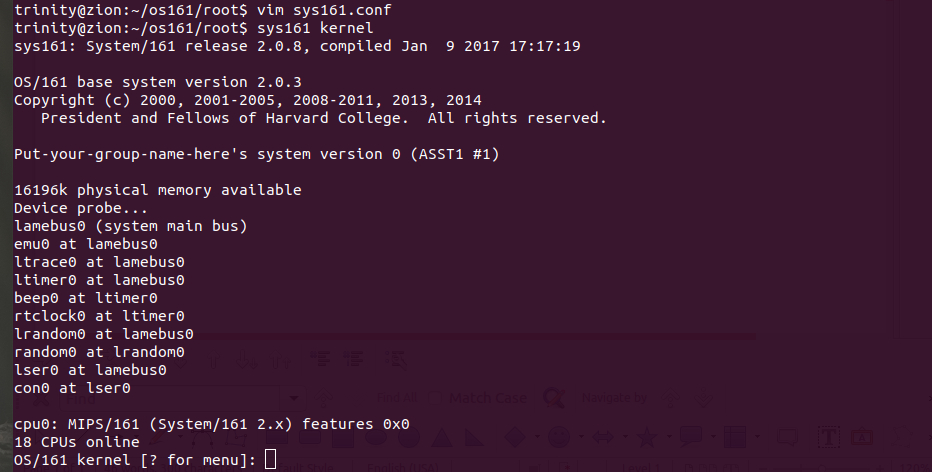
* Copyright (c) 2000, 2001-2005, 2008-2011, 2013, 2014

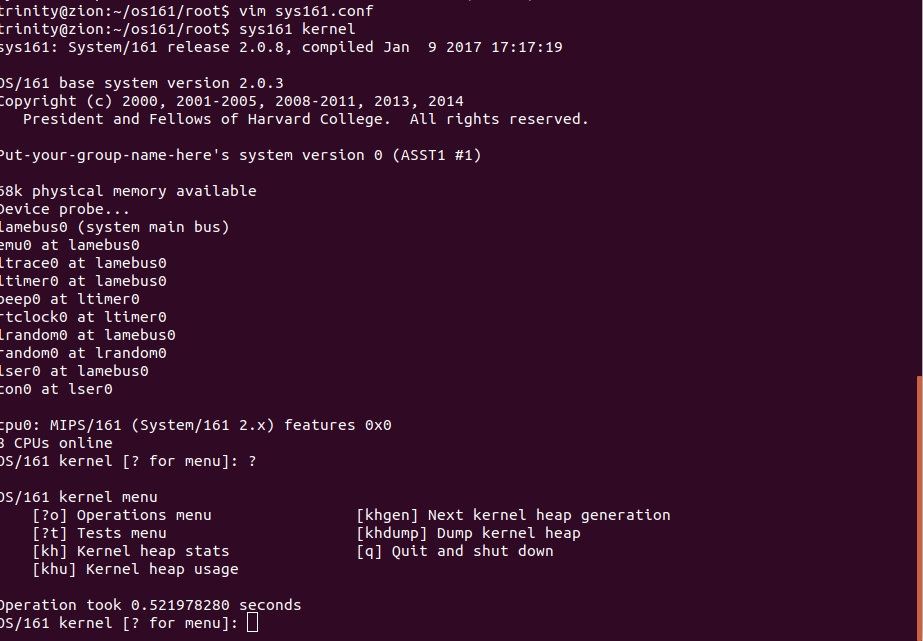
President and Fellows of Harvard College. All rights reserved.

* 16196k physical memory available

8 CPUs (CPU cores) online

* ~/os161/root/sys161.conf
* It seemed to boot a lot faster with 18 cores.

The system booted with 256K memory only has 68K left free



**Question 4**

**/os161/kern/arch/mips/include/kern/regdefs.h**

This file contains register labels for OS161.

**Question 5**

**~/src/os161/kern/include/spl.h**

The “spl()” functions set the priority level for the processor. The first, spl0() sets the IPL to 0, enabling all interrupts, while splhigh() sets the IPL to the highest value, disabling all interrupts.

**Question 6**

**~/os161/root/include/stdint.h (included in types.h)**

This is because the primitive type int can be different sizes depending on the system and OS. Setting a constrained size makes the code more portable (and is necessary to run on some architectures).

Reference:<https://stackoverflow.com/questions/13764892/what-determines-the-size-of-integer-in-c>

**Question 7**

**./os161/kern/include/device.h**

This file has a lot of structs/prototypes for devices.

**Question 8**

**./os161/kern/include/lib.h**

Using the DEBUG() macro

#define DEBUG(d, ...) ((dbflags & (d)) ? kprintf(\_\_VA\_ARGS\_\_) : 0)

**Question 9**

**os161/kern/include/synch.h**

Semaphores, locks, conditional variables, Reader-Writer locks

Reference: <https://stackoverflow.com/questions/8017507/definition-of-synchronization-primitive>

**Question 10**

**os161/kern/include/thread.h**

A thread\_yield causes a thread to wait until the next runnable thread runs, while a thread\_sleep causes a thread to wait a specified amount of time.

**Question 11**

**/os161/kern/thread/threadlist.c**

This describes a circular, doubly-linked list

**Question 12**

**/os161/kern/thread/thread.c**

(Zombies are threads that have exited but still

need to have thread\_destroy called on them.)

**Question 13**

These functions are used to copy memory chunks from kernel space into user space, and vise versa.

This protects against the kernel accessing a portion of memory that is not currently mapped by the MMU. This is useful whenever a user program provides a memory location for the kernel.

Reference: <https://en.wikipedia.org/wiki/Page_fault>

**Question 14**

**./os161/kern/dev/generic/console.c**

The function getch() will call getch\_intr(), which reads a character, while using interrupts to wait

for I/O completion

**Question 15**

**/os161/man/dev/null.html**

“The null device does nothing. Reads generate immediate EOF.”

**Question 16**

***os161*/kern/include/proc.h**

The process is a struct, which has a vnode pointing to the cwd. The lock for the struct is:

struct spinlock p\_lock.

**Question 17**

**/os161/kern/include/spl.h**

spl0(), splhigh(), and splx() are the functions to set the priority level, which enables or disables interrupts.

**Question 18**

***/os161*/kern/include/addrspace.h**

as\_copy(), “create a new address space that is an exact copy of an old one.”

**Question 19**

***/os161*//kern/thread/thread.c**

thread\_create() creates a new thread. That function additionally calls kmalloc(), thread\_machdep\_init(), and threadlistnode\_init().

**Question 20**

**/os161/kern/arch/mips/syscall/syscall.c**

What triggers a syscall in MIPS is having the callno register set (which I believe is $v0).

**Question 21**

**/os161/kern/arch/mips/syscall/syscall.c**

ENOSYS Function not implemented (POSIX.1-2001).

(from man page)

**Question 22**

**Question 23**

**/os161/kern/arch/mips/include/kern/regdefs.h**

#define sp $29 /\* stack pointer \*/

**Question 24**

***os161*/kern/include/spl.h**

The highest level is 1.

#define IPL\_HIGH 1

**Question 25**

***os161*/kern/include/clock.h**

/\* hardclocks per second \*/

#define HZ 100

**Question 26**

***/os161*//kern/include/kern/syscall.h**

#define SYS\_execv 2

**Question 27**

**Question 28**

**Question 29**

**Question 30**

**Question 31**

**Question 32**

**Question 33**

**Question 34**

**Question 35**

**Question 36**

**Question 37**