Slater Weinstock

Midterm

CS460

02/21/2019

Q1: With the modified KBD driver, P0 can NOT call kgetc().

EXPLAIN: WHY?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

kgetc has a sleep command and P0 cannot go to sleep.

---------------- ANSWER QUESTIONS Q2, Q3, Q4 below ---------------------

Given (1) to (6) in the ARM MTX code:

(1). Vector tabble at memory address 0

0x18: LDR PC, irq\_handler\_addr

irq\_handler\_addr: .word irq\_handler

(2). irq\_handler:

sub lr, lr, #4

stmfd sp!, {r0-r12, lr}

bl IRQ\_handler

ldmfd sp!, {r0-r12, pc}^

(3). IRQ\_handler{

if (VIC.statusRegBit32 && SIC.statusRefBit3)

kbd\_handler();

}

int hasData = 0; // global flag set to 0 in kbd\_init()

char c; // global char shared by KBD driver and processes

(4). kbd\_handler()

{

get scancode;

c = ASCII char mapped by scancode;

hasData = 1;

wakeup(&hasData);

}

(5). char kgetc()

{

if (hasData==0)

sleep(&hasData);

hasData = 0;

return c;

}

(6). Process\_Code()

{

unlock(); // allow CPU to accept IRQ interrupts

kgetc(); // Process P1 executes this line

}

============================================================================

Q2: Draw a diagram to show the control flow of the PROCESS P1

(7). Before any key is pressed

//while there is no data, P1 sleeps. P0 continually loops in the while(1) since readyQueue is empty.

(8). When a key is pressed.

//trigger interrupt A(in the vector table), calls interrupt handler , this will then wake up P1. Returns p1 to readyqueue, switches to p1, and

//do execution of p1

===========================================================================

Assume: P0 forked P1 and P2, and switch to run P1 (with P2, P0 in readyQueue).

When P1 tries to get a command line by

kgets(char line[ ]); // which calls kgetc() until '\r'

It would go to sleep on kgetc(), which switches to run P2, which also tries to

get a command line.

(9). Verify that each process will get a complete input LINE rather than a

different char of the same LINE.

Q3: EXPLAIN: HOW and WHY does (9) work? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

//P1 gets input line???? -->NEED TO CHECK

/\*Both P1 and P2 are woken up, P1 returns the char as c is a global. Once kgetc puts P1 to sleep,

resume P2, processes same c.

Q4: Interrupt handlers can only call wakeup() but should NEVER call sleep().

EXPALIN: WHY?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\*interrupts exist as something that can happen.

sleeping is related to a specific process. The interrupt

that is triggered will cause the current process to go to handle the interrupt/

If the interrupt handler sleeps, it would cause the interrupted process to sleep.

The interrupted process may not have to do with the interrupt reason MTX pag 283\*/