ECE 331

Homework 13

See course web site for due date

Place your typed homework answers in `vim`. Print single sided with your name using a **mono space font**. No need to restate questions. Fully investigating questions is required for a higher grade. Please use the kernel coding style for all code. Please use your RPi for developing answers. Although code should be written and run on a RPi, it should run on ANY POSIX compliant OS. As always, all code shall be comment, conform to the Linux Kernel Coding Style, and error conditions shall be checked and appropriately handled.

1. Complete the following php function that creates a table of size $w wide and $h high. For example if $w is 5 and $ h is 4, then the table would appears as follows:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

```
<?php
function table($w, $h)
{
        print "<TABLE BORDER=1\n>";


      print "</TABLE>\n";
}
?>
```

2. Calculate and give the missing addresses and netmasks. Show your work in each box.

Network (DDN)	Assignable IP (DDN)	Netmask (CIDR)	Broadcast (DDN)
	40.241.231.123	/13	
2.3.88.0			2.3.95.255

3. Write a python script that tests the Morse code kernel driver. Have the python script open and iterate over the dictionary (`/usr/share/dict/words` file, one word per line). Write each word to the Morse code kernel driver. Additionally print each word to `stdout`. Use of external commands, programs, and scripts not allowed. Not need for error checking.

4. Give the concise command to start two instances of the script written above
5. For each of the following questions, provide a concise answer with appropriate terminology.
   a) Explain why the `connect()` call requires a length argument passed.
   b) Explain why and when single core/single cpu systems require locking.
   c) Explain why the `printf()` function does exist in the kernel.
   d) Explain why error check and bounds checking is critical when writing kernel code.

6. Consider the following kernel code segment variant for our Morse kernel driver. It compiles.

```
... code ...

static ssize_t morse_write(struct file *filp, const char __user * buf, size_t count, loff_t * offp)
{
    char *dat_buf;
    struct mutex lock;

    init_mutex(&lock);
    if (mutex_is_locked(&lock)) {
        if (filp->f_flags&O_NONBLOCK) {
            return -EAGAIN;
        }
    }
    if (mutex_lock_interruptible(&lock)) {
        return -EINTR;
    }
    dat_buf=(char *)kmalloc(count,GFP_KERNEL);
    if (dat_buf==NULL) {
        mutex_unlock(&lock);
        return -ENOMEM;
    }
    if (copy_from_user(dat_buf,buf,count)) {
        mutex_unlock(&lock);
        return -EFAULT;
    }
... code ...
```

a) For the code segment above, will it always operate as expected? (state Yes or No)

b) Justify your answer above by showing, with examples, that is does or does not work. Give complete sentences. Correct keywords and terms expected.

c) If the code segment fails, explain how the code should be fixed. Give as much technical detail as you can. Correct keywords and terms expected.