// Author: Nicholas LaJoie
// ECE 331 - Homework #1
// Date: 1/25/17

1. The vim tutorial (vimtutor) has been completed!

2. The Linux Kernel Coding Style document has been thoroughly reviewed and will be conformed to.

3. C program passing all command line arguments:
    a. Source Code:

```
    // Author: Nicholas LaJoie
    // ECE 331 - Homework 1, Problem 3
    // Date: 1/25/17
    // Description: Program prints command line arguments passed by the user.

    #include <stdio.h>
    #include <string.h>

    int main(int argc, char * argv[])
    {
        int i;

        if (argc < 2) {
            printf("No arguments passed!\n");
            return -1;
        }
        for (i = 1; i < argc; i++) {
            printf("Argument: %s\n", argv[i]);
        }

        return 0;
    }
```

    b. `./prob3 "Game of Thrones \$eason 1 Episode 4.mp4" *\\\\\\\\*`

4. C program that determines memory speed of RPi 3:

```
// Author: Nicholas LaJoie
// ECE 331 - Homework 1, Problem 4
// Date: 1/25/17
// Description: Measure the speed (in nanoseconds) of memory for a buffer size passed as a
command line argument
// "inttypes" library and usage found at jhshi.me/2014/07/11/print-uint64-t-properly-in-c/

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>
#include <string.h>
#include <inttypes.h>

#define __STDC_FORMAT_MACROS
#define VALUE 17

int main(int argc, char * argv[])
{
        int *buffer, bufsize;
    struct timespec start, stop;

        if (argc < 2) {
        printf("No buffer size specified.\n");
        return -1;
        }
        if (argc > 2) {
        printf("Too many arguments!\n");
        return -2;
        }

    // Convert command line argument string to an int
        bufsize = atoi(argv[1]);
        if (bufsize <= 0) {
        printf("Invalid buffer size.\n");
        return -3;
        }
        buffer = (int *)malloc(bufsize);
    if (buffer == NULL) {
        printf("Memory allocation failed.\n");
        return -4;
    }

    // Start timer
        clock_gettime(CLOCK_MONOTONIC, &start);

    memset(buffer, VALUE, bufsize);

    // Stop timer
        clock_gettime(CLOCK_MONOTONIC, &stop);

    free(buffer);

    // Output data to a file in the format: buffer size, start time, stop time
    FILE *f = fopen("hw1_data.csv", "a");
    if (f == NULL) {
        printf("File could not be opened!\n");
        return -5;
    }
    fprintf(f, "%d, %"PRIu64", %"PRIu64"\n", bufsize, (uint64_t)start.tv_nsec,
(uint64_t)stop.tv_nsec);
    fclose(f);

    // Print to the console (so Sheaff can see it)
    printf("Buffer Size: %d Start: %"PRIu64" Stop: %"PRIu64"\n", bufsize,
(uint64_t)start.tv_nsec, (uint64_t)stop.tv_nsec);
        return 0;
}
```

5. MATLAB source code:

```
% Author: Nicholas LaJoie
% ECE 331 - Homework 1, Problem 5
% Date: 1/26/17

%% Plot RPi Memory Speeds

% Read data
f = '/Users/NicholasRossLaJoie/Desktop/hw1_data.csv';
data = dlmread(f,',',0,0);
bufsize = data(:,1);
start = data(:,2);
stop = data(:,3);

% Plot speed (kb/s) vs. buffer size (int)
figure(1)
subplot
semilogx(bufsize, ((bufsize*4)/8000)/((stop-start)/1e9), 'LineStyle', 'None','Marker', '*')
title('Memory Speed vs. Data Size')
xlabel('Buffer Size, Int'); % 4 Bytes per Int
ylabel('Speed, kb/s');
grid on;
```

6. FHS:

        a. The purpose of the /sys directory is to contain information about connected hardware devices. It utilizes the virtual file system from the 2.6 kernel ('sysfs') and displays device information in a hierarchical manner. (source: centos.org/docs/5/html/Deployment_Guide-en-US/s1-filesystem-fhs.html)

b. The requirements for /usr/lib: It is designed to hold libraries that are not intended to be directly used by the user or shell scripts. Different applications use a single subdirectory under '/usr/lib', where all architecture-dependent data used by that application must be contained. Various applications, such as 'sendmail' or 'X11', require symbolic links to '/lib/[application]'. (source: pathname.com/fhs/pub/fhs-2.3.pdf)

7. Completed.

8. Commands:
   a. `alias ll='ls -alF --color'`
   b. `ls -aX`
   c. `mv x /tmp/y`
   d. `rm *{10..25}*`