ECE 331

Homework 1 Solutions

1. Done.

2. Done.

3.

a)
```
// A. Sheaff 1/20/2017
// A program to print command line arguments
#include <stdio.h>
#include <stdlib.h>

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

   for (i=0;i<argc;i++) {
         printf("ARGV[%d]: %s\n",i,argv[i]);
   }

   return 0;
}
```

b)
```
# ./args 'Game of Thrones $eason 1 Episode 4.mp4' *\\\\\\\\*
```

4.
```
      // A Sheaff 1/23/2017
// A program to test memory speed
#include <stdio.h>
#include <malloc.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>
#include <time.h>

int main(int argc, char *argv[])
{
      size_t size;      // Buffer size
      unsigned char *b; // Buffer
      int ret;          // Return values
      struct timespec res;    // Timer resolution
      struct timespec s;      // Start time stamp
      struct timespec e;      // End time stamp

      // Pass the size of the buffer in bytes
      if (argc!=2) {
            printf("Usage: %s size\n",argv[0]);
            return 1;
      }
      // Print the size
      printf("%s ",argv[1]);

      // Get the resolution just to check it
      ret=clock_getres(CLOCK_MONOTONIC,&res);
      if (ret<0) {
```

```
            perror("getres");
            return 6;
      }
      printf("%lu %lu",res.tv_sec,res.tv_nsec);

      // Convert passed size to integer
      size=atoi(argv[1]);
      if (size<=0) {
            printf("buffer size is non-zero and positive\n");
            return 8;
      }

      // Allocate the memory and check for errors
      b=(unsigned int *)malloc(size);
      if (b==NULL) {
            perror("malloc");
            return 2;
      }

      // Start time stamp, write, and stop time stamp
      ret=clock_gettime(CLOCK_MONOTONIC,&s);
      if (ret<0) {
            perror("getres");
            return 6;
      }
      memset(b,'a',size);
      //sleep(100);
      ret=clock_gettime(CLOCK_MONOTONIC,&e);
      if (ret<0) {
            perror("getres");
            return 6;
      }
      printf(" %lu %lu %lu %lu\n",s.tv_sec,s.tv_nsec,e.tv_sec,e.tv_nsec);

      // clean up
      free(b);

      return 0;
}
```

5. Script to run the tests named regress

```
./mem 1
./mem 10
./mem 100
./mem 1000
./mem 10000
./mem 100000
./mem 1000000
./mem 10000000
./mem 100000000
```

Run the tests

```
# source regress > out
```

MATLAB m-file

```
load data
s=data(:,1);
start=data(:,4)+data(:,5)/1e9;
e=data(:,6)+data(:,7)/1e9;
dt=e-start;
rate=s./dt;
semilogx(s,rate/1024/1024)
title('RPi memory speed')
xlabel('data size, B')
```

```
ylabel('Rate, MB/s')
grid
```

6.
   a) Not found and not required
   b) No requirement are needed.

7. Done

8.
   a) `alias ll=’ll -alF --color’`
   b) `ls -X`
   c) `mv x /tmp/y`
   d) `rm -f *[1-9][0-9]* *2[0-5]*`