Blackjack

Q) How would you use rand() and srand() exactly to generate random numbers?

A) Please try reading the below links:
http://www.cplusplus.com/reference/cstdlib/srand/
http://www.cplusplus.com/reference/cstdlib/rand/
https://en.wikipedia.org/wiki/Random_seed
The former two describe the uses of srand and rand, and the last explains what a random seed is in a pseudorandom number generator.

Exif Viewer

Q) I want to be able to view the binary file to get a better feel of how it looks like.

A) Since binary files are not viewable by human eyes, there are several programs available in linux that display binary files in hexadecimal format. One of them is the 'xxd' program. You can invoke it as such:
xxd ./img1.jpg
If it's hard to view all at once in your terminal, we have learned how to redirect the stdout as such:
xxd ./img1.jpg > ./img1.hex
Now you can open the img1.hex file on your favorite editor and view it at your leisure. Try following the hexadecimal output while reading the EXIF format description.

Q) Would I have to do big-endian to little-endian conversion for this project?

A) No. A full implementation would require you to do conversion based on byte 12 of the EXIF header (II or MM), but I have made the job easier for you such that you only have to return an error if you read "MM". Since Intel ISA(Instruction Set Architecture) that the thoth machine supports is little-endian, and you are only required to parse little-endian files, you don't have to worry about this aspect except to check the endianness string (II or MM). There is one place where big-endian is always used in the 'length of APP1 block' field in the EXIF header, but you will not need that value in your program.

Q) When I read in the records or the headers into structs, should I use a certain data type?

A) You should decide on a data type that corresponds to the size and type of the field. For example, if the field is a string, then the data type should be a character array. If the field is a number, it should be one of the integer data types. To decide on which one of them to use, you can refer to the table on one of the data type slides, but it is best to try printing the size of the data type yourself using sizeof() to be sure.

Q) Should I worry about field alignment and padding when I read in the structs?

A) Normally yes. But in this case, the EXIF file format designers were very careful in making sure that the file records are perfectly aligned. For example, you will notice that 4-byte numbers are aligned at 4-byte boundaries and 2-byte numbers are aligned at 2-byte boundaries, such that the compiler would never have to generate padding to align accesses.

Q) I am completely lost. How do I go about reading in the individual bytes of the EXIF records?

A) Here is some example code to get you started:

```
#include <stdio.h>

// In your exif viewer, you should be reading the EXIF header all at once
// using a struct with members that correspond to the fields of the header.
// This example reads in the fields one by one just for demonstration.

int main(int argc, char **argv)
{
  unsigned short num;
  char str[5];
  FILE *f = fopen(argv[1], "r");
  if(fread(&num, 2, 1, f) != 1) { // Start of file marker
    fprintf(stderr, "Error while reading file marker.\n");
    return 1;
  }
  printf("%hx\n", num); // Looks flipped because of little-endian order of the thoth machine
  if(fread(&num, 2, 1, f) != 1) {// APP1 marker
    fprintf(stderr, "Error while reading APP1 marker.\n");
    return 1;
  }
  printf("%hx\n", num);
  if(fread(&num, 2, 1, f) != 1) { // Length of the APP1 block
    fprintf(stderr, "Error while reading length of APP1 block.\n");
    return 1;
```

```
  }
  printf("%hu\n", ntohs(num)); // ntohs converts from network byte order (big-endian) to host byte order (little-endian)
  if(fread(str, 1, 4, f) != 4) { // Exif string
    fprintf(stderr, "Error while reading Exif string.\n");
    return 1;
  }
  str[4] = '\0'; // Attach the null character for printing
  printf("%s\n", str);
  return 0;
}
```

Q) I am trying to read in the TIFF tags using a struct but somehow I am not getting the correct data read in. What am I doing wrong?

A) Most probably, you declared the struct with the wrong types. Try doing sizeof(struct StructName) to determine that the size of your struct coincides with the number of bytes for the TIFF tag (12 bytes) given.

Q) How do I jump back to reading the next TIFF tag, after having read an ASCII string type value by following the offset?

A) You need to either keep track of the current offset in the file you are reading, or an easier way is to use the ftell() file function. Try doing "man ftell" in the thoth machine to see how to use it. It returns the current file offset that you can store in a variable, and you can later use that offset to fseek back to that offset after reading in the string.