CSE 333 13su Exercise 2

out: Friday June 28, 2013
due: Monday, July 1, 2013 by **9:00 am**.

Your job is to write a C function that prints, in hex, the values of the bytes allocated to some variable. The function's prototype is

```
void DumpHex(void* pData, int byteLen);
```

and its output looks like this:

```
The 4 bytes starting at 0x7fff1081856C are: ff 01 30 4e
```

You should use this main function in the program you submit:

```
int main(int argc, char **argv) {
  char     charVal = '0';
  int32_t  intVal = 1;
  float    floatVal = 1.0;
  double   doubleVal  = 1.0;

  typedef struct {
    char     charVal;
    int32_t  intVal;
    float    floatVal;
    double   doubleVal;
  } Ex2Struct;
  Ex2Struct structVal = { '0', 1, 1.0, 1.0 };

  DumpHex(&charVal, sizeof(char));
  DumpHex(&intVal, sizeof(int32_t));
  DumpHex(&floatVal, sizeof(float));
  DumpHex(&doubleVal, sizeof(double));
  DumpHex(&structVal, sizeof(structVal));

  return EXIT_SUCCESS;
}
```

To do this, your implementation of DumpHex will need to do a few things:

- convince the compiler to let you access bytes in memory, starting from a `void*`
- use a loop to iterate once for each byte in the variable, using the pointer to extract that byte and print it.

- figure out how to use format specifiers to printf in order to print out an `uint8_t` in lower-case hexadecimal, using exactly two digits. As a hint, take inspiration from this code:

```
#include <inttypes.h>
...
  uint8_t a_byte = 0xD1;
  printf("The byte is: %02" PRIx8 " -- enjoy!\n", a_byte);
...
```

- figure out how to print a pointer value.

  Your code should produce output that is identical to the following, except for values that might change from run to run because of randomness outside your control. That randomness includes addresses and the values of uninitialized data.

```
$bash gcc -Wall -std=gnu99 -g -o ex2 ex2.c
$bash ls
ex2 ex2.c
$bash ./ex2
The 1 bytes starting at 0x7fff3c84a1ff are: 30
The 4 bytes starting at 0x7fff3c84a1f4 are: 01 00 00 00
The 4 bytes starting at 0x7fff3c84a1f8 are: 00 00 80 3f
The 8 bytes starting at 0x7fff3c84a1e8 are: 00 00 00 00 00 00 f0 3f
The 24 bytes starting at 0x7fff3c84a1d0 are: 30 b0 f0 00 01 00 00 00 00 00 80
3f 00 00 00 00 00 00 00 00 00 00 f0 3f
```

  (Note that the last output line is actually a single line, but probably wraps in your browser. Also, note that the number of bytes shown on the last line is not the sum of the numbers shown on the previous lines.)

Your code must:

- compile without errors or warnings on CSE Linux machines (lab workstations, attu, or CSE home VM)
- have no crashes, memory leaks, or memory errors on CSE linux machines
- be contained in a single file called "ex2.c" that compiles with the command "gcc -Wall -g -std=gnu99 -o ex2 ex2.c" -- do not submit a Makefile.
- be pretty: the formatting, modularization, variable and function names, and so on must make us smile rather than cry. (Suggestion: check your code with `clint` to see if it discovers any problems.)
- be robust: you should think about handling bogus input from the user, and you should handle hard-to-handle cases (if there are any) gracefully.
- have a comment at the top of your .c file with your name, student number, and CSE or UW email address.

  You should submit your exercise using the assignment dropbox linked on the main course web page.