15-668: Introduction to CUDA
Lecture 2 Notes

Magic File I/O Code:

```
std::ifstream amsnd("amsnd.txt");
std::stringstream ss;
ss << amsnd.rdbuf();
const char * const_str = ss.str().c_str();
unsigned int str_len = ss.str().length();

// copy to a mutable string
char host_str[str_len + 1];
memcpy(host_str, const_str, (str_len + 1) * sizeof(char));
```

Useful Functions:

cutil_inline.h:

cudaMalloc	Allocate device memory.
cudaMemcpy	Copy memory.

cuda_helpers.hh:

CHECK_FOR_CUDA_ERROR	Check for a cuda error.

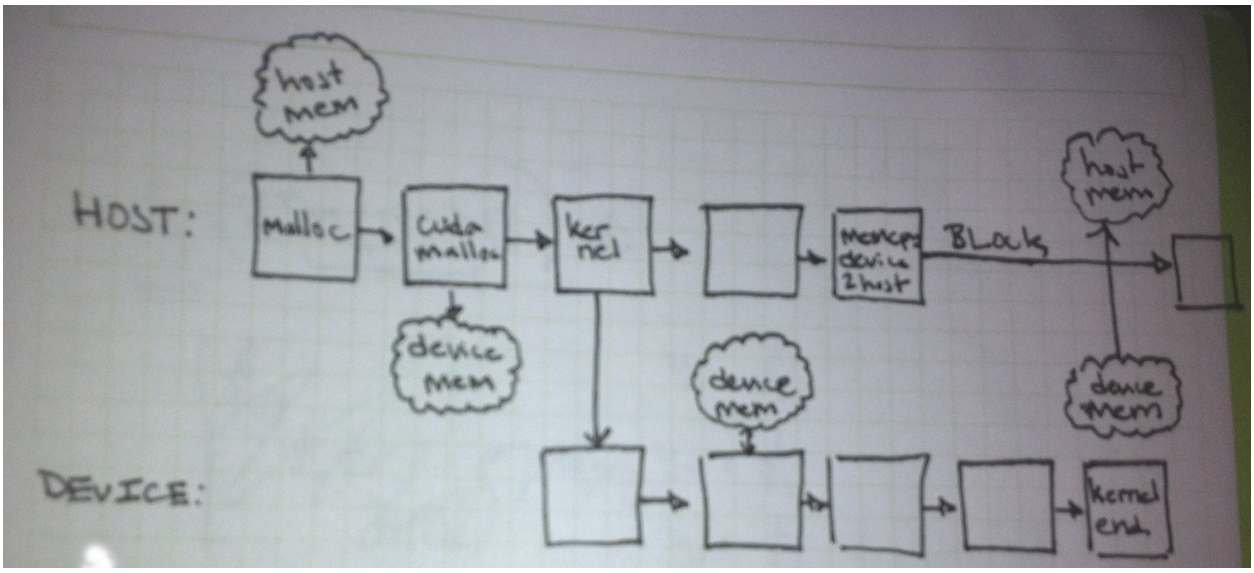
Capitalizing a string on the CPU:

```
// convert this to upper case
for (int ii = 0 ; ii < ss.str().length() ; ii++) {
     if (host_str[ii] >= 'a' && host_str[ii] <= 'z') {
          host_str[ii] -= a_to_A_distance;
     }
}
host_str[str_len] = '\0'; // not really necessary
```

Capitalizing a string in CUDA:

```
__global__
void capitalizeKernel(char  *str) {
     unsigned int ii = threadIdx.x;
     char cc = str[ii];
     if (cc >= 'a' && cc <= 'z') {
          cc -= a_to_A_distance;
     }
     str[ii] = cc;
}

...
capitalizeKernel<<<1., str_len>>>(device_str);
...
```



Full Capitalization Code:

```
#include <fstream>
#include <sstream>
#include <iostream>
#include <assert.h>

#include <cutil_inline.h>
// #include "cuda_helpers.hh"

const int difference = ((int) 'a') - ((int) 'A');
static const unsigned int n_threads_per_block = 512;

__global__
void capitalize(char * str, unsigned int str_len)
{
   const unsigned int ii = blockIdx.x * n_threads_per_block + threadIdx.x;
   if (ii >= str_len)
      return;
   char cc = str[ii];
   if (cc >= 'a' && cc <= 'z') { // if the caracter is lower case
      cc -= difference;          // capitalize it
      str[ii] = cc;
   }
}

int main(int argc, char **argv) {
   std::ifstream input("big_file.txt");
   std::stringstream ss;
   ss << input.rdbuf();
   unsigned int str_len = ss.str().length();

   // copy to a mutable string
   char host_str[str_len + 1];
   memcpy(host_str, ss.str().c_str(), (str_len + 1) * sizeof(char));

   ////////////////////
   // GPU CAPITALIZE //
   ////////////////////

   char * device_str = 0;
   cudaMalloc((void **) &device_str, (str_len + 1) * sizeof(char));
   cudaMemcpy(device_str, host_str, (str_len + 1) * sizeof(char),
   cudaMemcpyHostToDevice);

   unsigned int n_blocks = str_len / n_threads_per_block;
```

```
    if (str_len % n_threads_per_block != 0) {
       n_blocks++;
    }

    dim3 n_blocks_dim3(n_blocks, 1, 1);
    dim3 block_dim(n_threads_per_block, 1, 1);
    capitalize<<<n_blocks_dim3, block_dim>>>(device_str, str_len);

    // CHECK_FOR_CUDA_ERROR();

    cudaMemcpy(host_str, device_str, (str_len + 1) * sizeof(char),
    cudaMemcpyDeviceToHost);

    cudaFree(device_str);

    // output this string
    std::ofstream output("out.txt");
    output << host_str;
    output.close();

    // let's be good c++ programmers
    return 0;
}
```