LibGPUTK

- LibGPUTK
  - Introduction
  - Compilation
  - Docs & Usage
    - Command line arguments and IO types
    - Docs
      - gpuTKLog
        - Exmaples:
      - gpuTKTime_start
        - Exmaples:
      - gpuTKTime_stop
        - Examples
      - gpuTKArg_read
        - Examples

Introduction

LibGPUTK is a library developed by UIUC for and required by the NVIDIA GPU Teaching Kit. This library allows the GPU Teaching Kit to be self contained in the sense it does not need external libraries to run the laboratories.

The main features of the library are:

- Logging capabilities for debuging, including the production of a JSON file with a detailed log of a lab run
- Time measurement functions
- Import and export PPM images
- Import and export raw vectors and matrices

Compilation

We strongly recommend building the library using the instructions in the section ***Compiling and Running Labs*** in the **README.md** file on the root of the Teaching Kit Bitbucket repository.

Docs & Usage

Command line arguments and IO types

Programs using LibGPUTK like the GPU Teaching Kit labs, have the ability to read command line arguments. A typical command line run is:

```
./program -e <expected output file> -i <input file 1>,<input file 2> -o <output> -t <ty
```

Where:

- `-e <expected output file>` typically specifies the output produced by the data generator in the lab.
- `-i <input file 1>,<input file 2>` the input files or input parameters to be used in the lab.
- `-o <output>` file to store the output on certain labs.
- `-t <type>` type of the output file, it can be any of:
  - `vector`
  - `matrix`
  - `image`

Observe that arguments of a same option are specified without spaces and separeted by commas:

```
./program -i file1, file2  // Wrong usage, will result in an error or segfault
./program -i file1,file2   // Correct usage
```

Docs

gpuTKLog

The function **gpuTKLog** outputs a message to stdout and logs the message with a TAG.

```
gpuTKLog(TAG, T msg0, ... ):
  TAG: indicates the type of the message, useful for debugging purposes
  msg0: message to output to stdout and log on the TAG logs, where T is a printable typ
  ...:  additional messages to output to stdout and log on the TAG logs
```

Available TAG kinds are:

```
FATAL       // For reporting fatal errors
ERROR       // For reporting errors
WARN        // For reporting warnings
INFO        // For providing information
DEBUG       // For debugging
TRACE       // For traditional stdout
```

In order to output the detailed JSON log the user needs to set the flag `BUILD_JSONLOG` when building the library using the instructions in Compilation section.

Exmaples:

```
 int a = 0;
 int error = 1;
 gpuTKLog(TRACE, "Print value of a ", a);
 if(error != 0)
    gpuTKLog(ERROR, "An error has occurred, the error code is ", error);
```

gpuTKTime_start

The function **gpuTKTime_start** starts a timer identified by a TAG and a message.

```
gpuTKTime_start(TAG, string identifier):
  TAG: indicates the kind of the timer
  identifier: message to identify the timer
```

Important: In order to stop a timer, *gpuTKTime_stop* must be called with same TAG and identifier as *gpuTKTime_start*.

Available TAG kinds are:

```
Generic          // Tag for generic time measurements
IO               // Tag for input output time measurements
GPU              // Tag for general GPU time measurements
Copy             // Tag for synchronic memory transfers time measurements
Driver           // Tag for driver functions time measurements
CopyAsync        // Tag for a-synchronic memory transfers time measurements
Compute          // Tag for kernel time measurements
CPUGPUOverlap    // Tag for CPU GPU overlapping time measurements
```

In order to output the timing records to stdout the user needs to set the flag `BUILD_LOGTIME` when building the library using the instructions in Compilation section.

Exmaples:

```
gpuTKTime_start(Generic, "Initializing memory"); // Start generic timer with
                                                 // identifier "Initializing memory"
// Initialize memory
gpuTKTime_stop(Generic, "Initializing memory"); // Stop generic timer
```

gpuTKTime_stop

The function **gpuTKTime_stop** starts a timer identified by a TAG and a message.

```
gpuTKTime_stop(TAG, string identifier):
  TAG: indicates the kind of the timer
  identifier: message to identify the timer
```

Important: In order to stop a timer, *gpuTKTime_stop* must be called with same TAG and identifier as *gpuTKTime_start*.

Available TAG kinds are:

```
Generic          // Tag for generic time measurements
IO               // Tag for input output time measurements
GPU              // Tag for general GPU time measurements
Copy             // Tag for synchronic memory transfers time measurements
Driver           // Tag for driver functions time measurements
CopyAsync        // Tag for a-synchronic memory transfers time measurements
Compute          // Tag for kernel time measurements
CPUGPUOverlap    // Tag for CPU GPU overlapping time measurements
```

In order to output the timing records to stdout the user needs to set the flag `BUILD_LOGTIME` when building the library using the instructions in Compilation section.

Examples

```
gpuTKTime_start(Compute, "Performing vector add"); // Start compute timer with
                                                   // identifier "Performing vector a
// Launch kernel performing vector add
gpuTKTime_stop(Compute, "Performing vector add");  // Stop compute timer
```

gpuTKArg_read

The function **gpuTKArg_read** reads and process the command line arguments passed to the program.

```
gpuTKArg_read(int argc, char **argv):
  argc: argument count
  argv: string array with the arguments
```

Examples

```
// Call program with: ./main -i file1,file2,file3
int main(int argc, char **argv) {
  gpuTKArg_t args;
  args = gpuTKArg_read(argc, argv);
  std::string arg0 = gpuTKArg_getInputFile(args, 0); // arg0 now contains "file1"
}
```