#pragma once

#ifdef \_\_cplusplus

#include <twml/defines.h>

#include <cstdint>

#include <cstddef>

#include <cstring>

namespace twml {

// A low-level binary Thrift writer that can also compute output size

// in dry run mode without copying memory. See also https://git.io/vNPiv

//

// WARNING: Users of this class are responsible for generating valid Thrift

// by following the Thrift binary protocol (https://git.io/vNPiv).

class TWMLAPI ThriftWriter {

protected:

bool m\_dry\_run;

uint8\_t \*m\_buffer;

size\_t m\_buffer\_size;

size\_t m\_bytes\_written;

template <typename T> inline uint64\_t write(T val);

public:

// buffer: Memory to write the binary Thrift to.

// buffer\_size: Length of the buffer.

// dry\_run: If true, just count bytes 'written' but do not copy memory.

// If false, write binary Thrift to the buffer normally.

// Useful to determine output size for TensorFlow allocations.

ThriftWriter(uint8\_t \*buffer, size\_t buffer\_size, bool dry\_run = false) :

m\_dry\_run(dry\_run),

m\_buffer(buffer),

m\_buffer\_size(buffer\_size),

m\_bytes\_written(0) {}

// total bytes written to the buffer since object creation

uint64\_t getBytesWritten();

// encode headers and values into the buffer

uint64\_t writeStructFieldHeader(int8\_t field\_type, int16\_t field\_id);

uint64\_t writeStructStop();

uint64\_t writeListHeader(int8\_t element\_type, int32\_t num\_elems);

uint64\_t writeMapHeader(int8\_t key\_type, int8\_t val\_type, int32\_t num\_elems);

uint64\_t writeDouble(double val);

uint64\_t writeInt8(int8\_t val);

uint64\_t writeInt16(int16\_t val);

uint64\_t writeInt32(int32\_t val);

uint64\_t writeInt64(int64\_t val);

uint64\_t writeBinary(const uint8\_t \*bytes, int32\_t num\_bytes);

// clients expect UTF-8-encoded strings per the Thrift protocol

// (often this is just used to send bytes, not real strings though)

uint64\_t writeString(std::string str);

uint64\_t writeBool(bool val);

};

}

#endif