CRC32C

build passing build failing

New file format authors should consider HighwayHash. The initial version of this code was extracted from LevelDB, which is a stable key-value store that is widely used at Google.

This project collects a few CRC32C implementations under an umbrella that dispatches to a suitable implementation based on the host computer's hardware capabilities.

CRC32C is specified as the CRC that uses the iSCSI polynomial in RFC 3720. The polynomial was introduced by G. Castagnoli, S. Braeuer and M. Herrmann. CRC32C is used in software such as Btrfs, ext4, Ceph and leveldb.

Usage

```
#include "crc32c/crc32c.h"

int main() {
  const std::uint8_t buffer[] = {0, 0, 0, 0};
  std::uint32_t result;

  // Process a raw buffer.
  result = crc32c::Crc32c(buffer, 4);

  // Process a std::string.
  std::string string;
  string.resize(4);
  result = crc32c::Crc32c(string);

  // If you have C++17 support, process a std::string_view.
  std::string_view string_view(string);
  result = crc32c::Crc32c(string_view);

  return 0;
}
```

Prerequisites

This project uses CMake for building and testing. CMake is available in all popular Linux distributions, as well as in Homebrew.

This project uses submodules for dependency management.

```
git submodule update --init --recursive
```

If you're using Atom, the following packages can help.

```
apm install autocomplete-clang build build-cmake clang-format language-cmake \
    linter linter-clang
```

If you don't mind more setup in return for more speed, replace `autocomplete-clang` and `linter-clang` with `you-complete-me`. This requires setting up ycmd.

```
apm install autocomplete-plus build build-cmake clang-format language-cmake \
    linter you-complete-me
```

Building

The following commands build and install the project.

```
mkdir build
cd build
cmake -DCRC32C_BUILD_TESTS=0 -DCRC32C_BUILD_BENCHMARKS=0 .. && make all install
```

Development

The following command (when executed from `build/`) (re)builds the project and runs the tests.

```
cmake .. && cmake --build . && ctest --output-on-failure
```

Android testing

The following command builds the project against the Android NDK, which is useful for benchmarking against ARM processors.

```
cmake .. -DCMAKE_SYSTEM_NAME=Android -DCMAKE_ANDROID_ARCH_ABI=arm64-v8a \
    -DCMAKE_ANDROID_NDK=$HOME/Library/Android/sdk/ndk-bundle \
    -DCMAKE_ANDROID_NDK_TOOLCHAIN_VERSION=clang \
    -DCMAKE_ANDROID_STL_TYPE=c++_static -DCRC32C_USE_GLOG=0 \
    -DCMAKE_BUILD_TYPE=Release && cmake --build .
```

The following commands install and run the benchmarks.

```
adb push crc32c_bench /data/local/tmp
adb shell chmod +x /data/local/tmp/crc32c_bench
adb shell 'cd /data/local/tmp && ./crc32c_bench'
adb shell rm /data/local/tmp/crc32c_bench
```

The following commands install and run the tests.

```
adb push crc32c_tests /data/local/tmp
adb shell chmod +x /data/local/tmp/crc32c_tests
adb shell 'cd /data/local/tmp && ./crc32c_tests'
adb shell rm /data/local/tmp/crc32c_tests
```