ProjectDb Tutorial

Requirements

- `cmake version >= 3.16.*`
- `gcc version >= 10.2.0` (This is needed for proper c++20 support)

Usage

To use `ProjectDb`, follow the following steps:

1. Clone the repo from ProjectDb into `<root>`
2. Run `make init_build && make projectdb`

   This will build a `libprojectdb.a` static library, which needs to be linked against the application that uses `ProjectDb`.
3. Set the c++ standard with `-std=gnu++20`
4. Add `<root>/include` to the include path, and `#include <projectdb/projectdb.h>` in places where `ProjectDb` is used
5. Add `libprojectdb.a` to the linker path, and add `-lprojectdb -lpthread` in at the end of link line

An example is given as following:

Example

The directory structure for this example is as following:

```
.
..
ProjectDbTest
    main.cpp -> This is the application that's trying to use ProjectDb
ProjectDb -> This is cloned ProjectDb repo
    cmake-build-release
        libprojectdb.a -> This is generated after running "make init_build && make projectdb"
```

`main.cpp`, which is the application, is as following:

```cpp
// main.cpp

#include "projectdb/projectdb.h"

int main() {
    // To customize db configs, create a config file based on <root>/config/config.template,
    // and initialize ProjectDb with:
    // projectdb::ProjectDb db {"<path_to_config_file>"};
    projectdb::ProjectDb db;
    db.set("Hello", "World!");
    return 0;
}
```

The command used to build `main.cpp` is as following:

```
g++ -std=gnu++20 -I../ProjectDb/include main.cpp -o main \
    -L../ProjectDb/cmake-build-release/ -lprojectdb -lpthread
```

Usage

There are three apis currently supported by `ProjectDb`:

1. `void set(const std::string& key, const std::string& value);`
2. `void remove(const std::string& key);`
3. `std::optional<std::string> get(const std::string& key);`

To use this api, first create a `ProjectDb` object, with an optional parameter of a config file. Then, use this object to call the above apis.

Note that with the current `ProjectDb` implementation, these apis are not thread safe. `ProjectDb` object needs to be locked by user if the apis are called in a multi-threaded fashion.

The current implementation doesn't provide multi-process support. And it is important to note that **only one `ProjectDb` should be created for one `DB_FILE_PATH`**, otherwise, there's a risk of data corruption.

Also, although some configs can be updated between runs, it's better to keep config unchanged after the database is created.