

Project2

milestone 1 - additional resource

Disk Space Manager

Notice

- This document is made to give you the entire structure of the project
- The following projects(2,3,4 ...) will be built on top of this structure
- All operations including testing and building libraries can be performed within this project base code
- Your submission should follow the rules stated on this document
- If you're done with milestone 1, start analyzing **bpt.c** and **bpt.h** for milestone 2, which will be announced in a close future

Download

- Download the ***db_project.tar.gz*** file from Piazza
 - It's on the Resources tab -> General Resources

The screenshot shows a dark-themed web browser window for the Piazza Class Profile of ITC 2038-13182. The navigation bar includes links for Google Drive, Maps, GitHub, Gitlab - SCS lab, and SCSSLAB JIRA. The main content area displays three sections: Homework Solutions, Lecture Notes, and General Resources.

Homework Solutions: A message states "Nothing has been added to the Homework Solutions section, yet." with a note to click "Add Links" or "Add Files".

Lecture Notes: A message states "Nothing has been added to the Lecture Notes section, yet." with a note to click "Add Links" or "Add Files".

General Resources: This section is titled "General Resources". It lists three files:

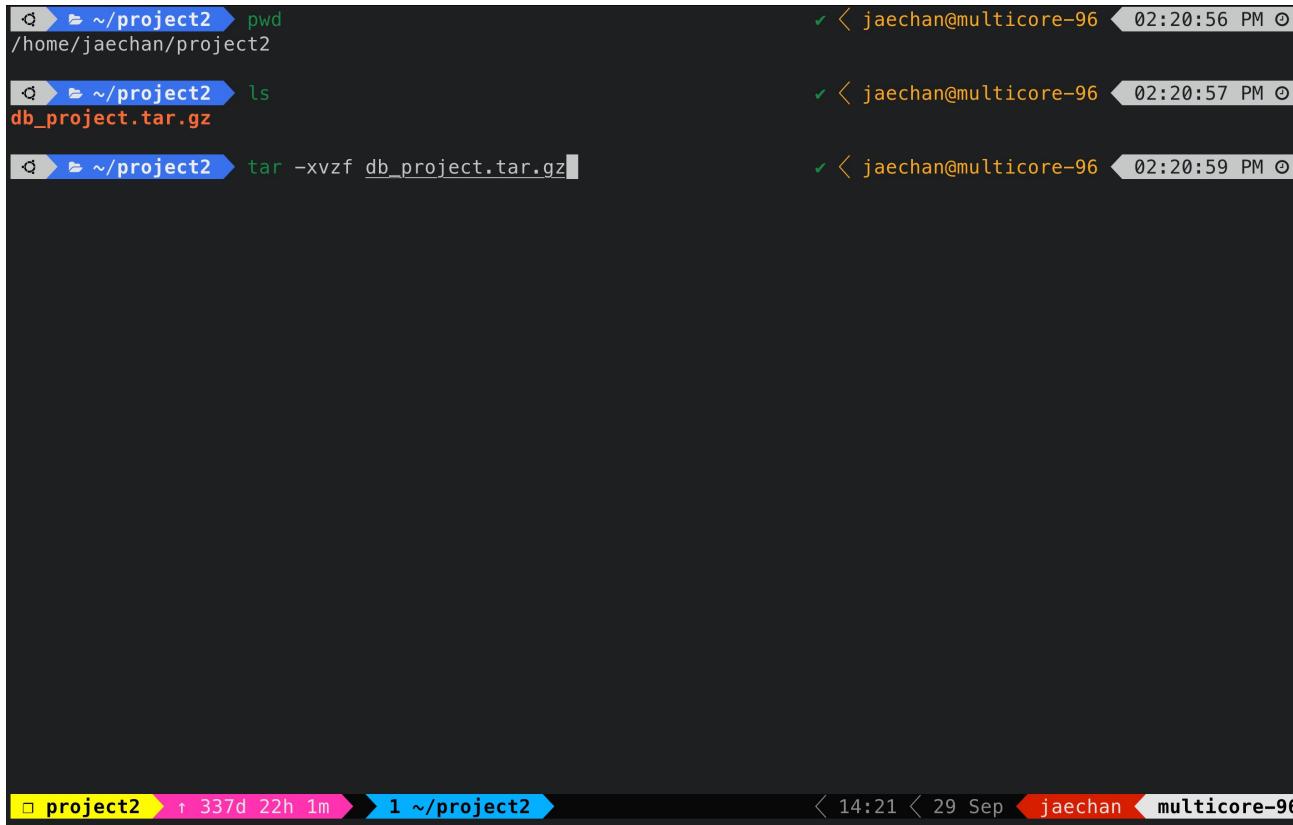
- db_project.tar.gz** (highlighted with a red box)
- Manage_project_with_GitLab.pdf**
- insert.txt**

Each file entry includes "Actions" buttons for Edit, Post a note, Update file, and Delete.

At the bottom of the page, a copyright notice reads: "Copyright © 2021 Piazza Technologies, Inc. All Rights Reserved."

Decompress

- Decompress the downloaded file using the given command
 - *tar -xvzf db_project.tar.gz*



A terminal session showing the decompression of a tar.gz file. The session starts with the user navigating to their project directory, listing files, and then decompressing the archive.

```
↳ ~project2 ➤ pwd  
/home/jaechan/project2  
↳ ~project2 ➤ ls  
db_project.tar.gz  
↳ ~project2 ➤ tar -xvzf db_project.tar.gz
```

The terminal window shows three command-line entries. The first entry shows the current directory as `/home/jaechan/project2`. The second entry lists the file `db_project.tar.gz`. The third entry runs the command `tar -xvzf db_project.tar.gz`. The terminal interface includes a status bar at the bottom with the current directory, date, time, and user information.

Project Structure

- You will have a directory named ***db_project***

```
↳ ~./project2 tar -xvzf db_project.tar.gz ✓ < jaechan@multicore-96 03:16:06 PM ◉
db_project/
db_project/test/
db_project/test/file_test.cc
db_project/test/CMakeLists.txt
db_project/test/basic_test.cc
db_project/DbConfig.h.in
db_project/.clang-format
db_project/db/
db_project/db/include/
db_project/db/include/file.h
db_project/db/include/bpt.h
db_project/db/src/
db_project/db/src/bpt.cc
db_project/db/src/file.cc
db_project/db/CMakeLists.txt
db_project/CMakeLists.txt
db_project/main.cc

↳ ~./project2 ls ✓ < jaechan@multicore-96 03:16:11 PM ◉
db_project db_project.tar.gz

↳ ~./project2 ls db_project ✓ < jaechan@multicore-96 03:16:13 PM ◉
CMakeLists.txt db DbConfig.h.in main.cc test

↳ ~./project2 █ ✓ < jaechan@multicore-96 03:16:14 PM ◉

↳ project2 ➜ 337d 22h 56m ➜ 1 ~./project2 ➜ < 15:16 < 29 Sep ➜ jaechan ➜ multicore-96
```

Project Structure

- The directory structure of the ***db_project*** directory is like below

```
q ➤ ~/project2 ➤ tree db_project
db_project
├── CMakeLists.txt
├── db
│   ├── CMakeLists.txt
│   ├── include
│   │   └── bpt.h
│   │       └── file.h
│   └── src
│       └── bpt.cc
│           └── file.cc
└── DbConfig.h.in
    └── main.cc
    └── test
        ├── basic_test.cc
        └── CMakeLists.txt
            └── file_test.cc

4 directories, 11 files

q ➤ ~/project2 ➤
```

Project Structure

```
ls ~/project2 db_project  
db_project  
├── CMakeLists.txt  
├── db  
│   ├── CMakeLists.txt  
│   ├── include  
│   │   └── bpt.h  
│   │   └── file.h  
│   └── src  
│       └── bpt.cc  
│           └── file.cc  
└── DbConfig.h.in  
main.cc  
└── test  
    ├── basic_test.cc  
    ├── CMakeLists.txt  
    └── file_test.cc  
  
4 directories, 11 files
```

- ***db_project***
 - Project directory
- ***db***
 - Library directory
 - ***libdb.a*** file is created by the codes in this directory
- ***test***
 - Test directory
 - Your test codes are placed here

Project Structure

```
➜ ~ ~/project2 ➜ tree db_project
db_project
├── CMakeLists.txt
└── db
    ├── CMakeLists.txt
    ├── include
    │   └── bpt.h
    │   └── file.h
    └── src
        └── bpt.cc
        └── file.cc
├── DbConfig.h.in
└── main.cc

test
└── basic_test.cc
└── CMakeLists.txt
└── file_test.cc

4 directories, 11 files
```

- ***db_project/CMakeLists.txt***
 - This file orchestrates the entire project's build process
 - In normal circumstances, you won't have to edit this file
- ***db_project/main.cc***
 - The main function for interactively testing your B+tree is located here
 - We **do not** use this file on evaluation
 - Use it for your convenience only
- ***db_project/DbConfig.h.in***
 - No need to touch this

Project Structure

```
ls ~/project2
db_project
├── CMakeLists.txt
└── db
    ├── CMakeLists.txt
    └── include
        ├── bpt.h
        └── file.h
    └── src
        ├── bpt.cc
        └── file.cc
└── DbConfig.h.in
└── main.cc
└── test
    ├── basic_test.cc
    ├── CMakeLists.txt
    └── file_test.cc

4 directories, 11 files
```

- ***db/CMakeLists.txt***

- You **MUST** edit this file if you add any source/header files for your library

- ***db/include***

- You **MUST** put your headers inside the include directory

- ***db/src***

- You **MUST** put your sources inside the source directory

- You may create more directories within the ***include,src*** directory

Project Structure

```
d/CMakeLists.txt ➤ t/CMakeLists.txt > buffers
1 # Sources
2 set(DB_SOURCE_DIR src)
3 set(DB_SOURCES
4   ${DB_SOURCE_DIR}/bpt.cc
5   ${DB_SOURCE_DIR}/file.cc
6   # Add your sources here
7   # ${DB_SOURCE_DIR}/foo/bar/your_source.cc
8
9 # Headers
10 set(DB_HEADER_DIR include)
11 set(DB_HEADERS
12   ${DB_HEADER_DIR}/bpt.h
13   ${DB_HEADER_DIR}/file.h
14   # Add your headers here
15   # ${DB_HEADER_DIR}/foo/bar/your_header.h
16 )
17
18 add_library(db STATIC ${DB_HEADERS} ${DB_SOURCES})
19
20 target_include_directories(db
21   PUBLIC "${CMAKE_CURRENT_SOURCE_DIR}/${DB_HEADER_DIR}"
22 )
23

NORMAL ➤ db/CMakeLists.txt cmake ➤ utf-8[unix] ➤ 4% ↵:q ↵project2 ➤ 337d 22h 33m ➤ 1 exit ➤ 14:53 < 29 Sep ➤ jaechan ➤ multicore-96
```

Add your source/header(s) here
(db/CMakeLists.txt)

Project Structure

```
ls ~/project2 tree db_project
db_project
├── CMakeLists.txt
├── db
│   ├── CMakeLists.txt
│   ├── include
│   │   └── bpt.h
│   │   └── file.h
│   └── src
│       └── bpt.cc
│           └── file.cc
└── DbConfig.h.in
└── main.cc
└── test
    ├── basic_test.cc
    ├── CMakeLists.txt
    └── file_test.cc

4 directories, 11 files
```

- ***test/CMakeLists.txt***

- The contents we saw in the GoogleTest tutorial are here
- You **MUST** edit this file if you add any source files for your tests
- This file links your library to the test code

- ***test/file_test.cc***

- You **MUST** put your disk space manager test codes in here

- ***test/basic_test.cc***

- The tutorial code (*IsPrime, Factorial*)

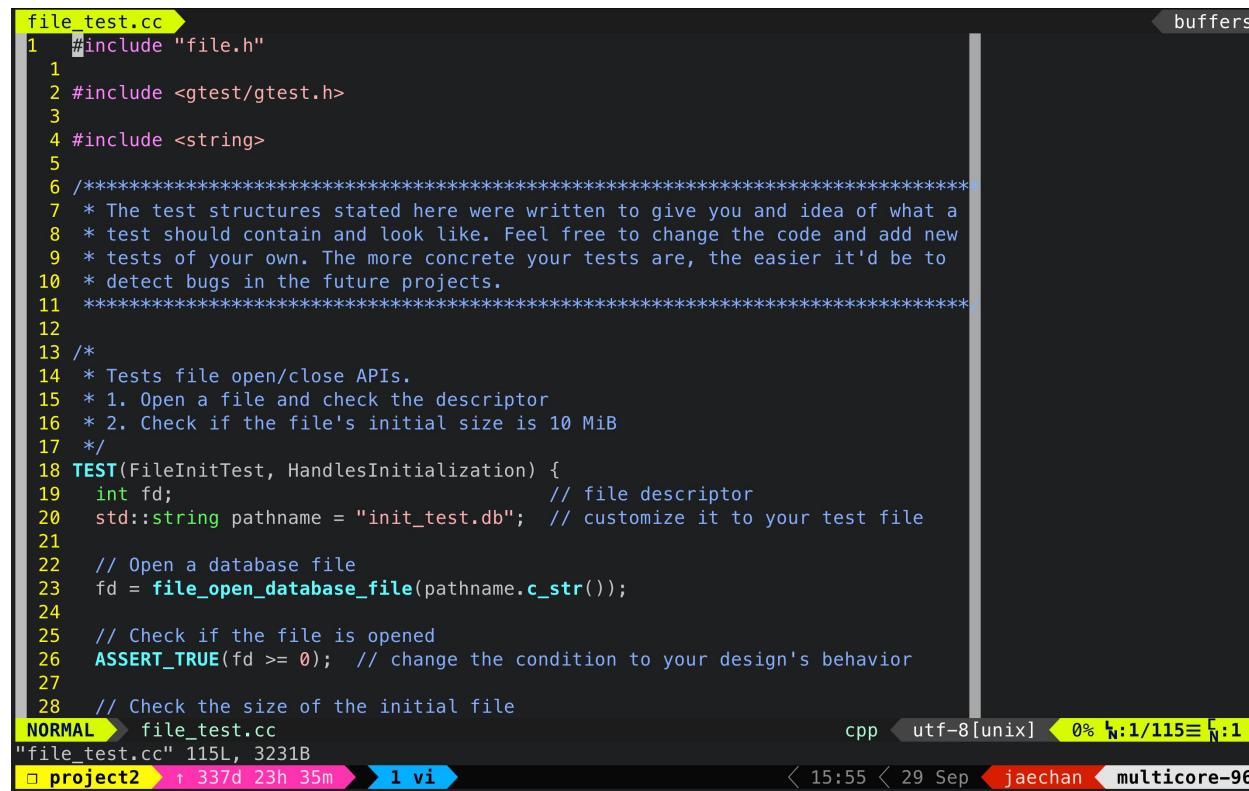
Project Structure

```
t/CMakeLists.txt ➤ buffers
1 # GoogleTest
1 include(FetchContent)
2 FetchContent_Declare(
3   googletest
4   URL https://github.com/google/googletest/archive/e4717df71a4f45bf9f0ac88c6cd9846a0bc248dd.zip)
5
6 set(gtest_force_shared_crt ON CACHE BOOL "" FORCE)
7 FetchContent_MakeAvailable(googletest)
8
9 set(DB_TESTS
10   file_test.cc
11   basic_test.cc
12   # Add your test files here
13   # foo/bar/your_test.cc
14 )
15
16 add_executable(db_test ${DB_TESTS})
17
18 target_link_libraries(
19   db_test
20   db
21   gtest_main
22 )
23
24 include(GoogleTest)
25 gtest_discover_tests(db_test)
26
~ ~
NORMAL ➤ test/CMakeLists.txt cmake ➤ utf-8[unix] 3% ↵:1/27 ≡ ↵:1
"test/CMakeLists.txt" 27L, 506B
project2 ➤ ↑ 337d 22h 57m ➤ 1 vi < 15:17 < 29 Sep ➤ jaechan ➤ multicore-96
```

Add your source/header(s) here
(test/CMakeLists.txt)

Project Structure

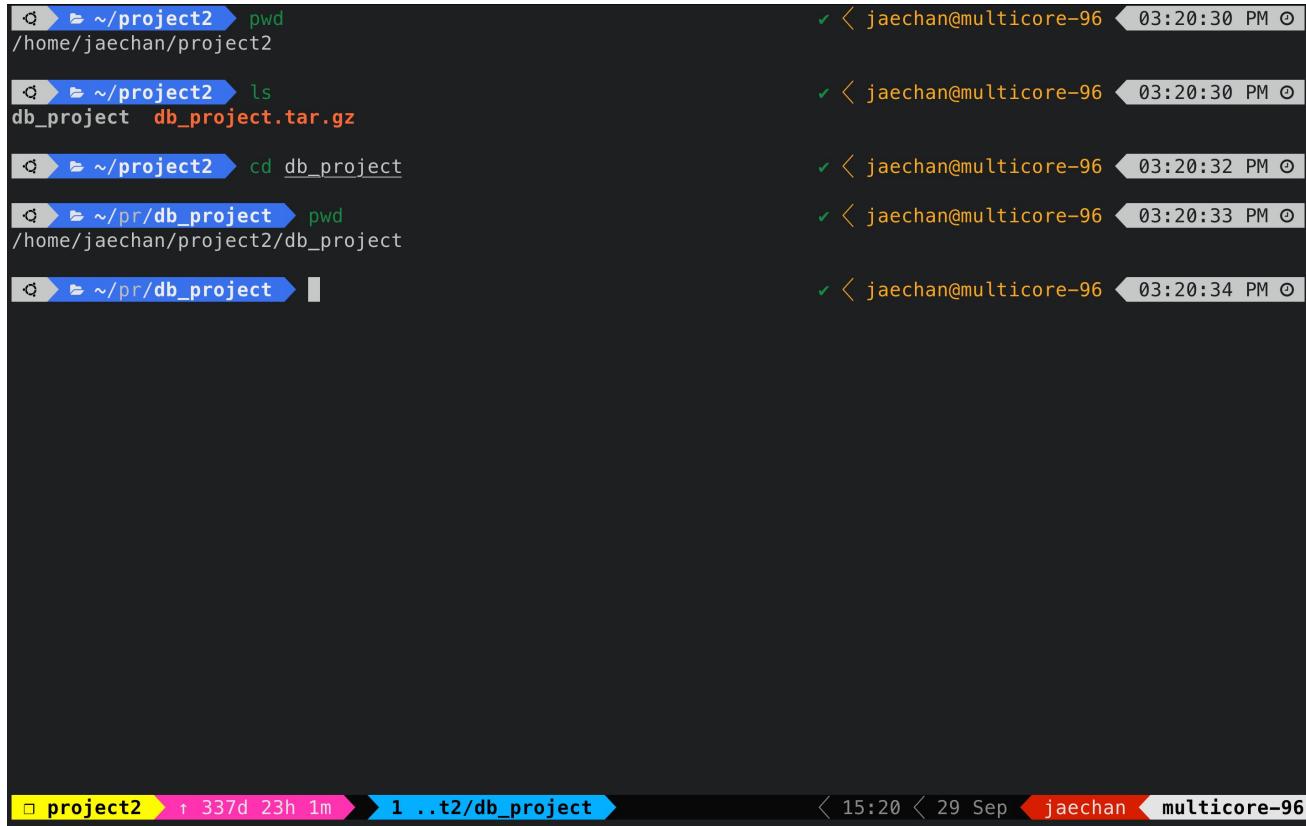
- We have a sample skeleton code just for reference
 - *test/file_test.cc*
- It's just for reference and you should implement your own test codes



```
file_test.cc
1 #include "file.h"
2
3 #include <gtest/gtest.h>
4
5 #include <string>
6
7 //*****
8 * The test structures stated here were written to give you an idea of what a
9 * test should contain and look like. Feel free to change the code and add new
10 * tests of your own. The more concrete your tests are, the easier it'd be to
11 * detect bugs in the future projects.
12 //*****
13 /*
14 * Tests file open/close APIs.
15 * 1. Open a file and check the descriptor
16 * 2. Check if the file's initial size is 10 MiB
17 */
18 TEST(FileInitTest, HandlesInitialization) {
19     int fd;                                // file descriptor
20     std::string pathname = "init_test.db"; // customize it to your test file
21
22     // Open a database file
23     fd = file_open_database_file(pathname.c_str());
24
25     // Check if the file is opened
26     ASSERT_TRUE(fd >= 0); // change the condition to your design's behavior
27
28     // Check the size of the initial file
NORMAL  file_test.cc
"file_test.cc" 115L, 3231B
project2  1337d 23h 35m  1 vi
buffers
cpp  utf-8[unix] 0% 1/115 1
< 15:55 < 29 Sep jaechan multicore-96
```

Build

- Move into the project directory (*db_project*)
 - *cd db_project*



The screenshot shows a terminal window with a dark background and light-colored text. It displays a sequence of commands and their outputs:

- 1. `(pwd)`: Shows the current working directory is `/home/jaechan/project2`.
- 2. `(ls)`: Shows files `db_project` and `db_project.tar.gz`.
- 3. `(cd db_project)`: Changes the directory to `/home/jaechan/project2/db_project`.
- 4. `(pwd)`: Shows the current working directory is `/home/jaechan/project2/db_project`.

On the right side of the terminal, there is a vertical timeline with green checkmarks and timestamps indicating the progression of the session:

- 03:20:30 PM: Initial connection.
- 03:20:30 PM: Listing files.
- 03:20:32 PM: Entering the project directory.
- 03:20:33 PM: Confirming the new working directory.
- 03:20:34 PM: Final status update.

At the bottom of the terminal, there is a navigation bar with icons for file operations and a footer with system information:

- project2
- ↑ 337d 23h 1m
- 1 ..t2/db_project
- < 15:20 < 29 Sep jaechan multicore-96

Build

- Execute the cmake command we used in the GoogleTest tutorial
 - *cmake -S . -B build*

The screenshot shows a terminal window with a dark background and light-colored text. At the top, it displays the command: `cmake -S . -B build`. Below this, the terminal outputs the results of the configuration process, including compiler detection (GNU 10.3.0 for C and CXX), feature detection for C and CXX compilers, and the finding of Python 3.6.9. It also shows the search for pthread.h and the configuration of threads. The output ends with the message: "Build files have been written to: /home/jaechan/project2/db_project/build". The terminal window includes standard Linux navigation keys at the bottom and a status bar at the bottom right showing the date and time.

```
cmake -S . -B build
-- The C compiler identification is GNU 10.3.0
-- The CXX compiler identification is GNU 10.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: /usr/bin/python3.6 (found version "3.6.9") found components: Interpreter
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Failed
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Configuring done
-- Generating done
-- Build files have been written to: /home/jaechan/project2/db_project/build
```

Build

- Execute the cmake command we used in the GoogleTest tutorial
 - *cmake --build build*

```
~> ~pr/db_project > cmake --build build
[ 6%] Building CXX object db/CMakeFiles/db.dir/src/bpt.cc.o
[ 12%] Building CXX object db/CMakeFiles/db.dir/src/file.cc.o
[ 18%] Linking CXX static library ../lib/libdb.a
[ 18%] Built target db
[ 25%] Building CXX object CMakeFiles/disk_based_db.dir/main.cc.o
[ 31%] Linking CXX executable bin/disk_based_db
[ 31%] Built target disk_based_db
[ 37%] Building CXX object _deps/googletest-build/googletest/CMakeFiles/gtest.dir/src/gtest-all.cc.o
[ 43%] Linking CXX static library ../../lib/libgtest.a
[ 43%] Built target gtest
[ 50%] Building CXX object _deps/googletest-build/googletest/CMakeFiles/gtest_main.dir/src/gtest_main.cc.o
[ 56%] Linking CXX static library ../../lib/libgtest_main.a
[ 56%] Built target gtest_main
[ 62%] Building CXX object test/CMakeFiles/db_test.dir/file_test.cc.o
[ 68%] Building CXX object test/CMakeFiles/db_test.dir/basic_test.cc.o
[ 75%] Linking CXX executable ../bin/db_test
[ 75%] Built target db_test
[ 81%] Building CXX object _deps/googletest-build/googlemock/CMakeFiles/gmock.dir/src/gmock-all.cc.o
[ 87%] Linking CXX static library ../../lib/libgmock.a
[ 87%] Built target gmock
[ 93%] Building CXX object _deps/googletest-build/googlemock/CMakeFiles/gmock_main.dir/src/gmock_main.cc.o
[100%] Linking CXX static library ../../lib/libgmock_main.a
[100%] Built target gmock_main
~> ~pr/db_project >
```

Build

- Move into the build directory
 - *cd build*

```
| q ➜ ~/pr/db_project ➜ cd build          ✓ < jaechan@multicore-96 03:28:19 PM ⓘ
| q ➜ ~/pr/d/build ➜ ls                  ✓ < jaechan@multicore-96 03:28:21 PM ⓘ
bin           cmake_install.cmake   DartConfiguration.tcl _deps      test
CMakeCache.txt  compile_commands.json db          lib       Testing
CMakeFiles     CTestTestfile.cmake   DbConfig.h    Makefile
CTestTestfile.cmake

| q ➜ ~/pr/d/build ➜                   ✓ < jaechan@multicore-96 03:28:22 PM ⓘ

```

project2 ↑ 337d 23h 8m ➜ 1 ..project/build < 15:28 < 29 Sep jaechan multicore-96

Execute

```
q ➤ ~/pr/d/build ➤ ls
bin CTestTestfile.cmake lib
CMakeCache.txt DartConfiguration.tcl Makefile
CMakeFiles db test
cmake_install.cmake DbConfig.h Testing
compile_commands.json _deps

q ➤ ~/pr/d/build ➤ ls bin
db_test disk_based_db

q ➤ ~/pr/d/build ➤ ls lib
libdb.a libgmock_main.a libgtest_main.a
libgmock.a libgtest.a

q ➤ ~/pr/d/build ➤
```

- ***bin***

- The executable file (***disk_based_db***) for your ***main.cc*** file will be generated here
- Also, the test code will be built as an executable file (***db_test***) within here

- ***lib***

- Your library will be built within here as ***libdb.a***

Execute

- Try executing the executable files within the ***bin*** directory
 - ***./disk_based_db***

```
↳ ~pr/d/b/bin ➤ pwd ✓ jaechan@multicore-96 03:45:31 PM
/home/jaechan/project2/db_project/build/bin

↳ ~pr/d/b/bin ➤ ./disk_based_db ✓ jaechan@multicore-96 03:45:36 PM
bpt version 1.14 -- Copyright (C) 2010 Amittai Aviram http://www.amittai.com
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.

B+ Tree of Order 4.
Following Silberschatz, Korth, Sridharan, Database Concepts, 5th ed.

To build a B+ tree of a different order, start again and enter the order
as an integer argument: bpt <order> (3 <= order <= 20).
To start with input from a file of newline-delimited integers,
start again and enter the order followed by the filename:
bpt <order> <inputfile> .

Enter any of the following commands after the prompt > :
  i <k> -- Insert <k> (an integer) as both key and value.
  f <k> -- Find the value under key <k>.
  p <k> -- Print the path from the root to key k and its associated value.
  r <k1> <k2> -- Print the keys and values found in the range [<k1>, <k2>]
  d <k> -- Delete key <k> and its associated value.
  x -- Destroy the whole tree. Start again with an empty tree of the same order.
  t -- Print the B+ tree.
  l -- Print the keys of the leaves (bottom row of the tree).
  v -- Toggle output of pointer addresses ("verbose") in tree and leaves.
  q -- Quit. (Or use Ctl-D.)
  ? -- Print this help message.
> █

project2 337d 23h 26m ➤ 1 ./disk_based_db ➤ 15:46 29 Sep jaechan multicore-96
```

GoogleTest

- You can run your tests built with the **GoogleTest** library in two ways
 1. Run it as we learned before using the **ctest** command

```
靓 > ~ /pr/d/build > ctest
Test project /home/jaechan/project2/db_project/build
  Start 1: FileInitTest.HandlesInitialization
  1/10 Test #1: FileInitTest.HandlesInitialization ... Passed 0.00 sec
  Start 2: FileTest.HandlesPageAllocation
  2/10 Test #2: FileTest.HandlesPageAllocation ..... Passed 0.00 sec
  Start 3: FileTest.CheckReadWriteOperation
  3/10 Test #3: FileTest.CheckReadWriteOperation ..... Passed 0.00 sec
  Start 4: FactorialTest.HandlesZeroInput
  4/10 Test #4: FactorialTest.HandlesZeroInput ..... Passed 0.00 sec
  Start 5: FactorialTest.HandlesPositiveInput
  5/10 Test #5: FactorialTest.HandlesPositiveInput ... Passed 0.00 sec
  Start 6: PrimeTest.HandlesOneInput
  6/10 Test #6: PrimeTest.HandlesOneInput ..... Passed 0.00 sec
  Start 7: PrimeTest.HandlesPositiveInput
  7/10 Test #7: PrimeTest.HandlesPositiveInput ..... Passed 0.00 sec
  Start 8: VectorTest.HasSameContent
  8/10 Test #8: VectorTest.HasSameContent ..... Passed 0.00 sec
  Start 9: IntVectorTest.HasSameContent
  9/10 Test #9: IntVectorTest.HasSameContent ..... Passed 0.00 sec
  Start 10: IntVectorTest.IsNotNullInitially
  10/10 Test #10: IntVectorTest.IsNotNullInitially ... Passed 0.00 sec

100% tests passed, 0 tests failed out of 10

Total Test time (real) = 0.05 sec
靓 > ~ /pr/d/build >
靓 < jaechan@multicore-96 < 04:07:54 PM ○
靓 > ~ /pr/d/build > 1 ..project/build >
靓 < 16:07 < 29 Sep < jaechan < multicore-96
```

GoogleTest

- You can run your tests built with the **GoogleTest** library in two ways
 2. Run the executable file ***db_test*** within the ***bin*** directory (***./db_test***)

```
[  ↵  ~pr/d/b/bin  ↶ ./db_test          ✓ < jaechan@multicore-96 ◀ 03:47:04 PM ⏺ [16/1598]
Running main() from /home/jaechan/project2/db_project/build/_deps/googletest-src/googletest/src/gtest_main.cc
[=====] Running 10 tests from 6 test suites.
[-----] Global test environment set-up.
[-----] 1 test from FileInitTest
[ RUN   ] FileInitTest.HandlesInitialization
[     OK ] FileInitTest.HandlesInitialization (0 ms)
[-----] 1 test from FileInitTest (0 ms total)

[-----] 2 tests from FileTest
[ RUN   ] FileTest.HandlesPageAllocation
[     OK ] FileTest.HandlesPageAllocation (0 ms)
[ RUN   ] FileTest.CheckReadWriteOperation
[     OK ] FileTest.CheckReadWriteOperation (0 ms)
[-----] 2 tests from FileTest (0 ms total)

[-----] 2 tests from FactorialTest
[ RUN   ] FactorialTest.HandlesZeroInput
[     OK ] FactorialTest.HandlesZeroInput (0 ms)
[ RUN   ] FactorialTest.HandlesPositiveInput
[     OK ] FactorialTest.HandlesPositiveInput (0 ms)
[-----] 2 tests from FactorialTest (0 ms total)

[-----] 2 tests from PrimeTest
[ RUN   ] PrimeTest.HandlesOneInput
[     OK ] PrimeTest.HandlesOneInput (0 ms)
[ RUN   ] PrimeTest.HandlesPositiveInput
[     OK ] PrimeTest.HandlesPositiveInput (0 ms)
[-----] 2 tests from PrimeTest (0 ms total)

[-----] 1 test from VectorTest
[ project2  ↵  337d 23h 27m  ↶ 1 ..ect/build/bin  ↶ 15:47 < 29 Sep  ↶ jaechan ↶ multicore-96 ]
```

IMPORTANT NOTES

- The following conditions **MUST** be satisfied at all circumstances
 - There should be a static library file ***libdb.a*** located within the directory ***db_project/build/lib***, after you build your project
 - Your source/header files must be within the directory we specified before (***src, include***)
 - Within the corresponding directories, you may create more directories as you'd like
 - Your source file extension should be **.cc**
 - Your header file extension should be **.h**
 - Use cmake version \geq 3.15 (Prefer to use the latest cmake, version 3.21.3)
- Plagiarism is **STRICTLY FORBIDDEN** as mentioned before
 - We take this issue seriously, and we'll make sure that the student caught plagiarizing will pay the most significant price that we can give within the university's authority

IMPORTANT NOTES

- Make sure that your project builds on the following environment
 - Ubuntu 18.04
 - cmake (version \geq 3.15)
 - C++ 17

Thank you
