Project2
milestone 1 - additional resource

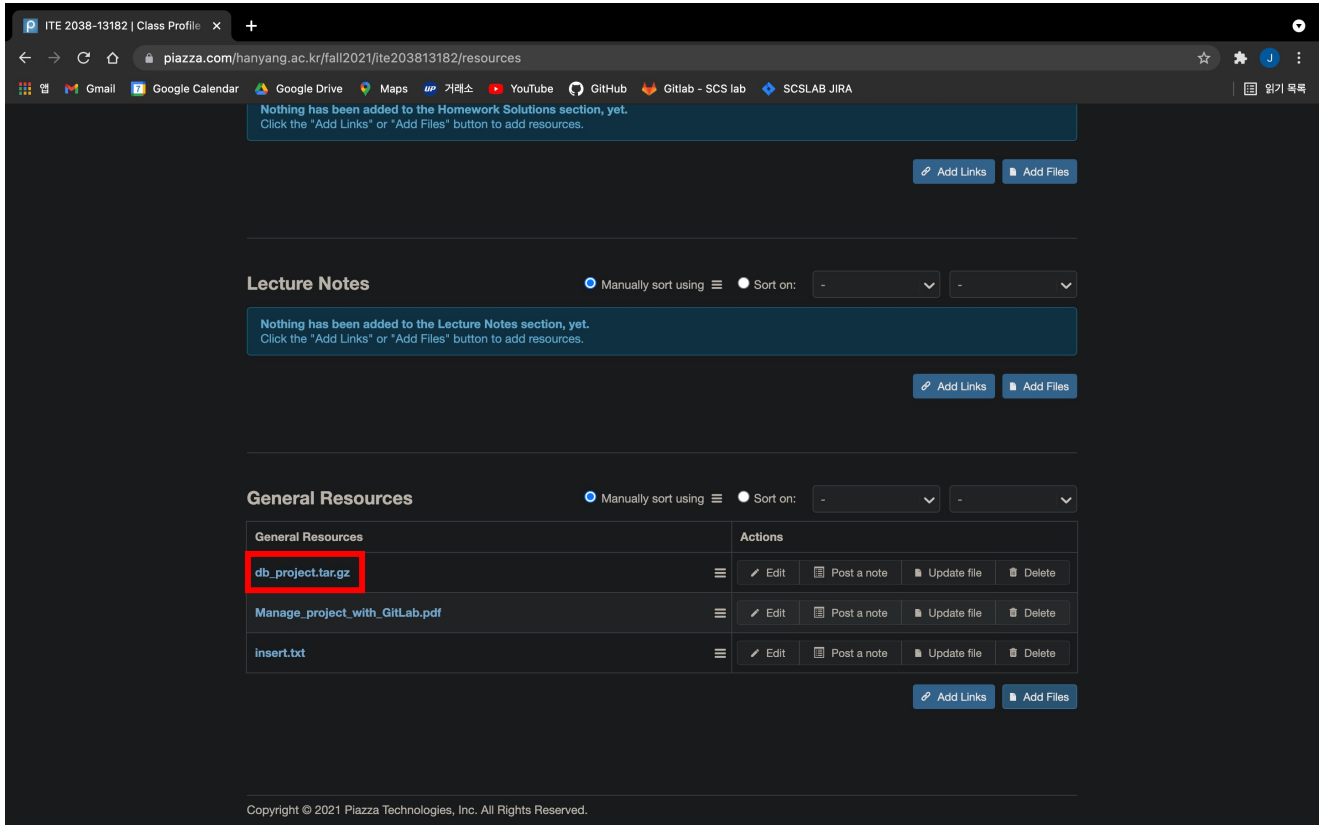
Disk Space Manager

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



Decompress

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

Project Structure

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

Project Structure

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

```
~/project2 tree db_project                                  jaechan@multicore-96 02:23:38 PM
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

~/project2                                                  jaechan@multicore-96 02:23:41 PM

project2 ↑ 337d 22h 4m 1 ~/project2                         14:23 29 Sep jaechan multicore-96
```

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
```

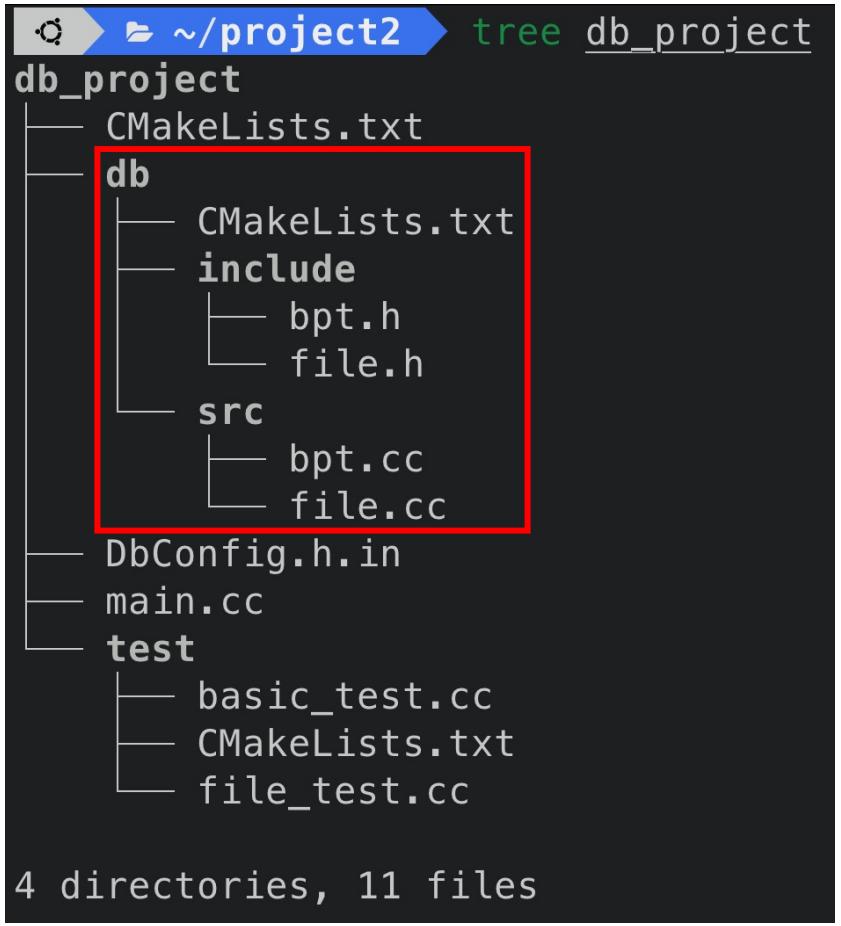
- ***db_project***
  - Project directory
- ***db***
  - Library directory
  - ***libdb.a*** file is created by the codes in this directory
- ***test***
  - Test directory
  - Your test codes are placed here

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
```

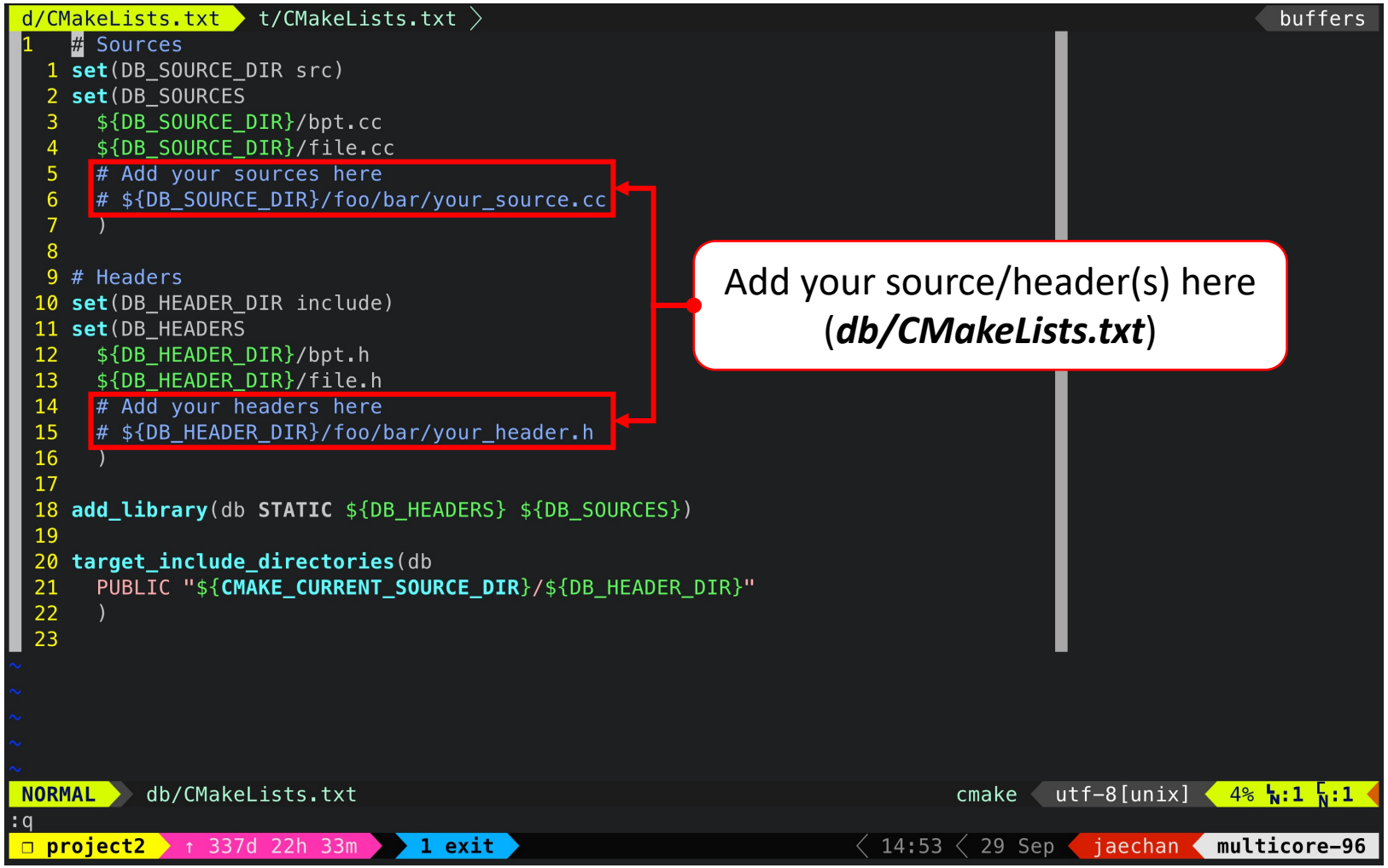
- ***db_project/CMakeLists.txt***
  - This file orchestrates the entire project's build process
  - In normal circumstances, you won't have to edit this file
- ***db_project/main.cc***
  - The main function for interactively testing your B+tree is located here
  - We **do not** use this file on evaluation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

Project Structure



- ***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



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
```

- ***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

```
1   # GoogleTest
  1 include(FetchContent)
  2 FetchContent_Declare(
  3   googletest
  4   URL https://github.com/google/googletest/archive/e4717df71a4f45bf9f0ac88c6cd9[illegible]46a0bc248dd.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
```

buffers

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

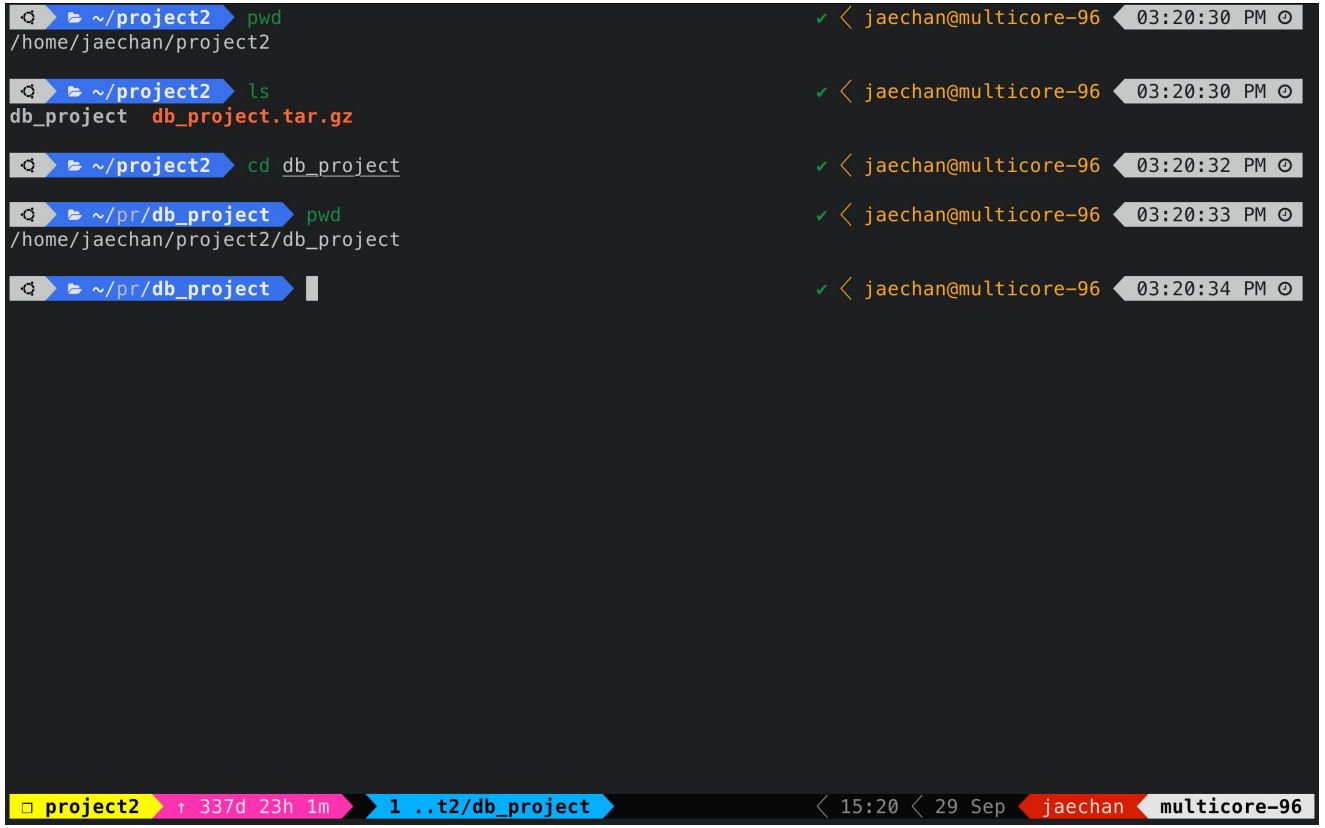
```
NORMAL  test/CMakeLists.txt                    cmake  utf-8[unix]  3%  N:1/27≡ N:1
"test/CMakeLists.txt" 27L, 506B
 project2  ↑ 337d 22h 57m  1 vi                〈 15:17 〈 29 Sep  jaechan  multicore-96
```

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

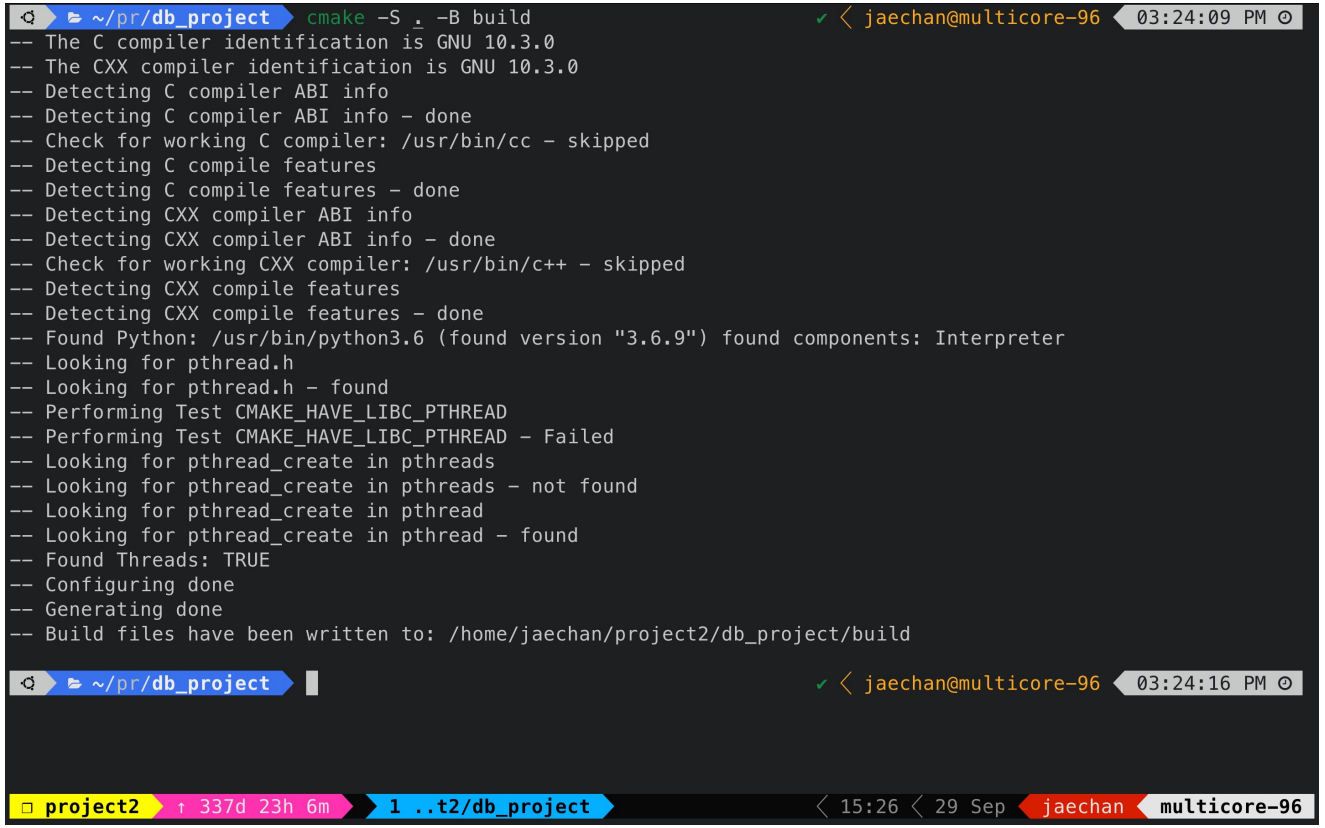
Build

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



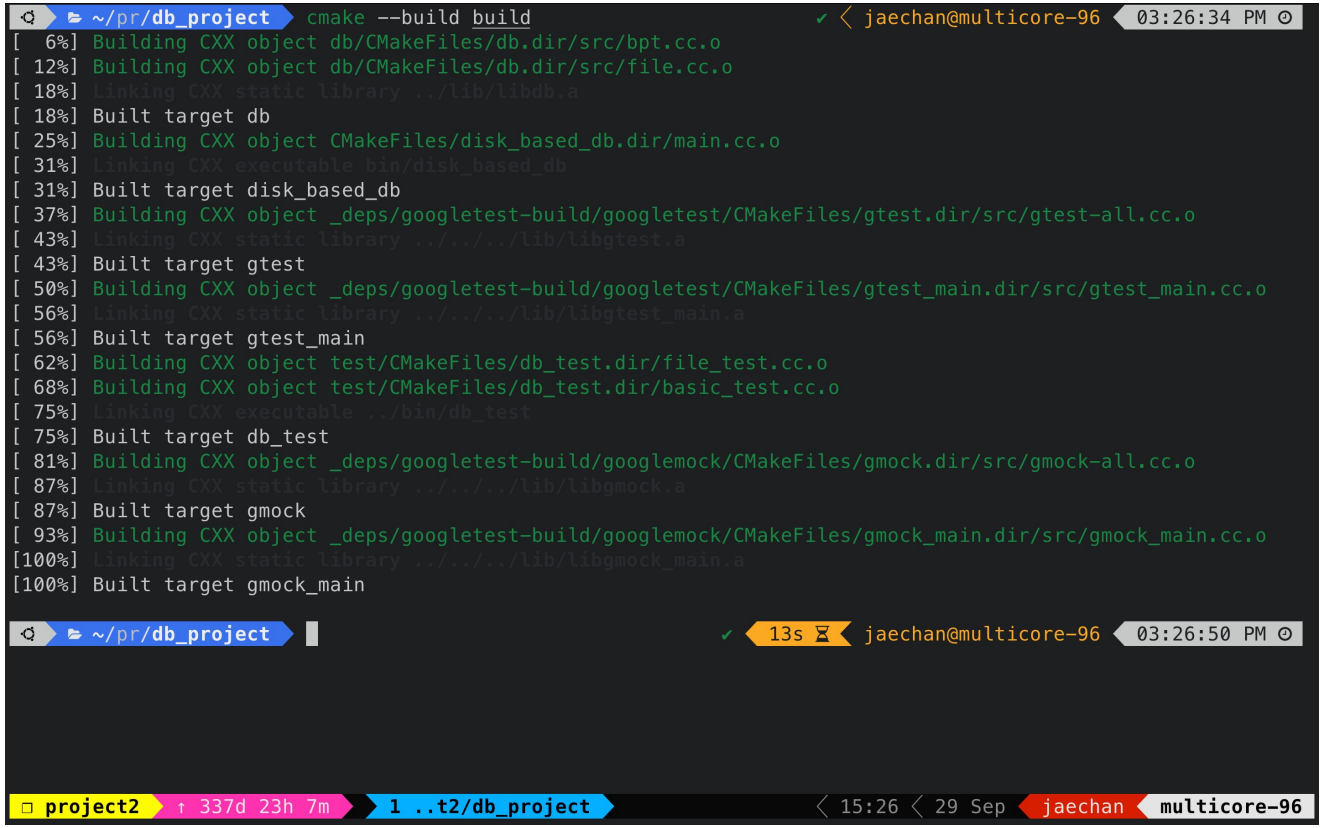
Build

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



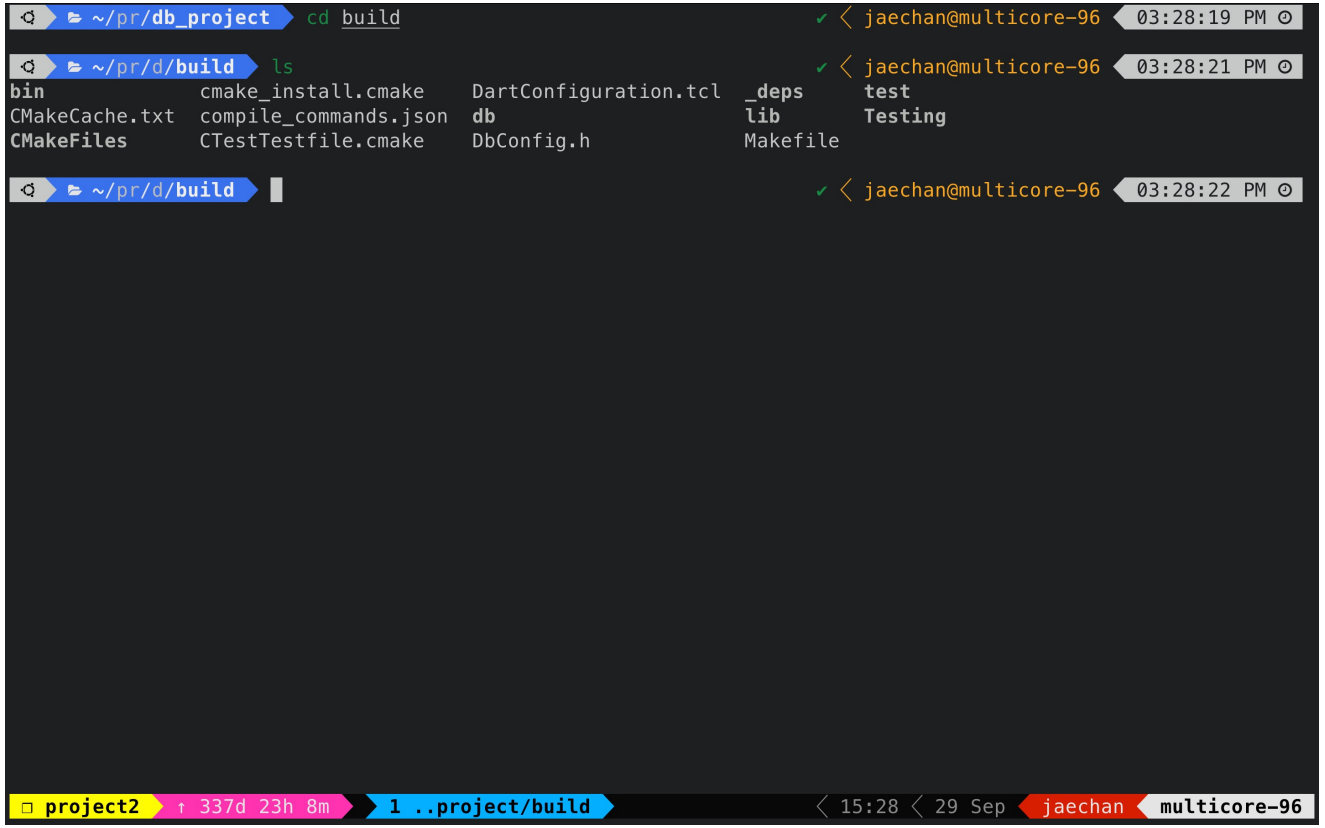
Build

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



Build

- Move into the build directory
  - *cd build*



Execute

```
~/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

~/pr/d/build ls bin
db_test  disk_based_db

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

~/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***

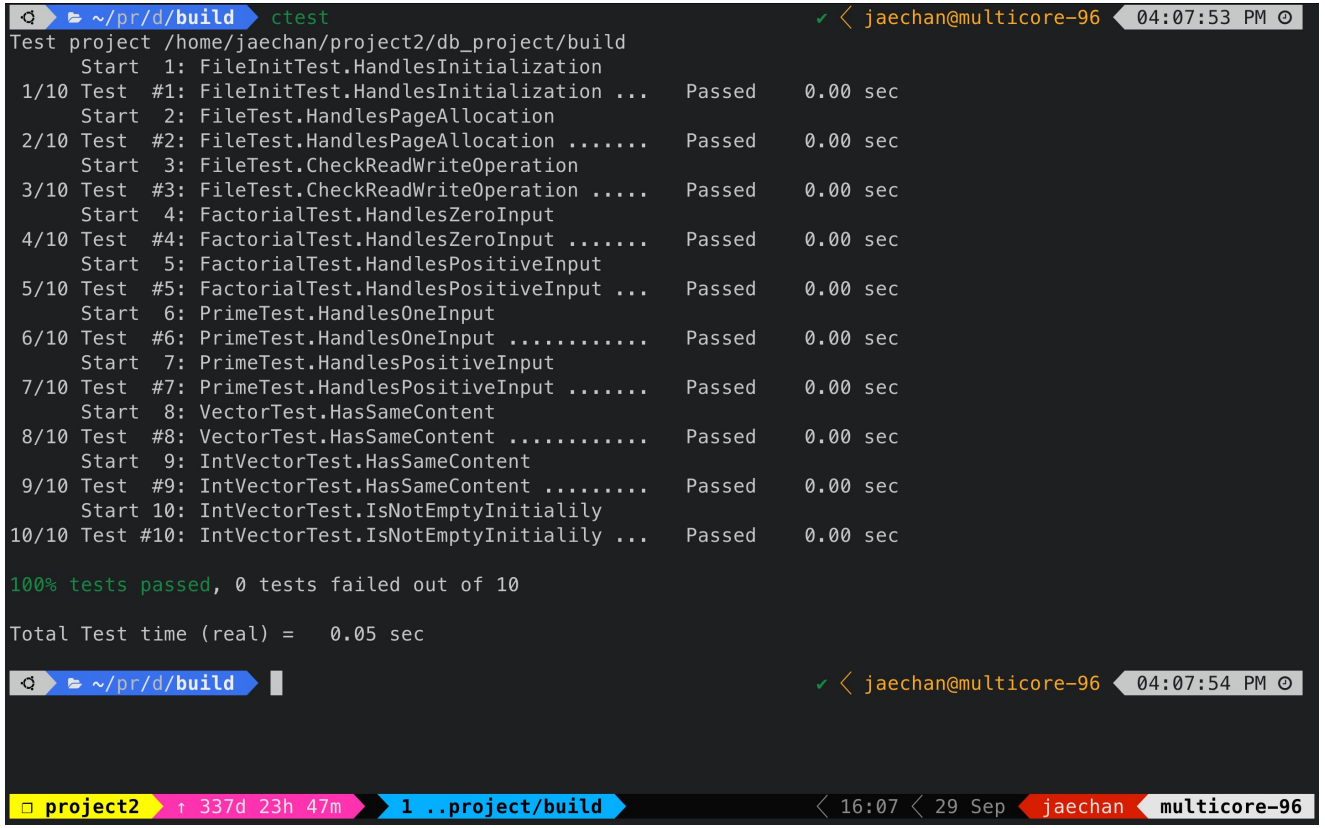
Execute

- 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, Sidarshan, 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
```

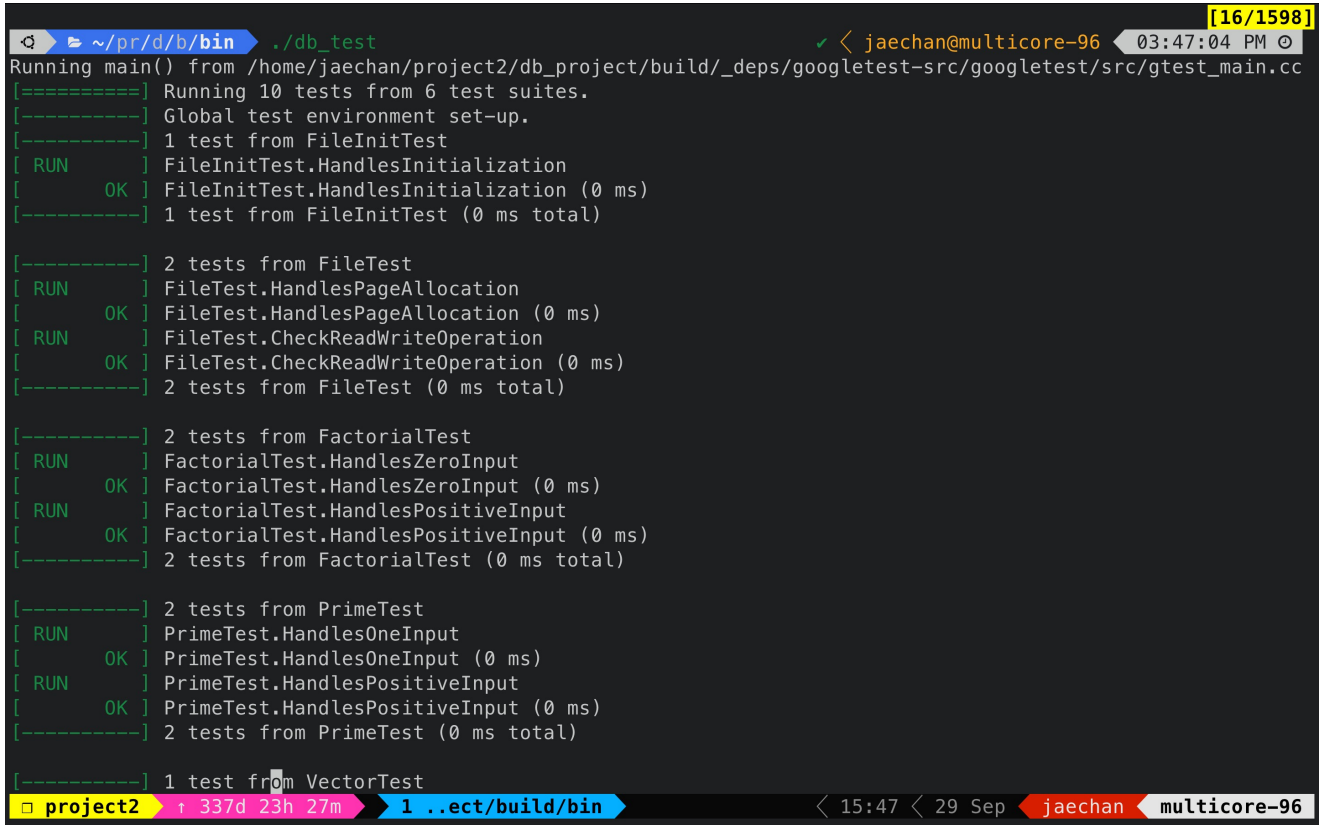
GoogleTest

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



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***)



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 >= 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 >= 3.15)
  - C++ 17

Thank you