Project Timing Demo

Fall 2021 - CS 2341

These are some notes about how the Project Check In and Timing Demos will run:

1. All the projects will be built and run from Dr. Fontenot's computer. You will be here to answer any questions that come up and to see it run for yourself.
2. What you need to have completed:
    a. AVL Tree as index for words from documents
    b. Document Parser. **You may include the stemming and stop word removal. Do some tests to see how they affect run time as the number of documents parsed increases.**
    c. Functionality to parse all documents in a given folder, path provided as a command line argument.
    d. Functionality to query the AVL tree based on 1 single word provided as a command line argument. You will print the document IDs that contain the word searched for.

3. During or before the the demo, I will:
    a. pull your code from github
    b. build it using cmake (as if it were being built in clion) in **Release** mode (See below for more info) and the compiler will support up to and including c++17. .
    c. execute the project on a small (~10) document data set as proof of concept.
    d. execute the project on increasingly large data sets as time permits
    e. Timing data will be gathered using the **time** command.

Your program should execute with exactly 2 command line arguments:

1. full path of folder to search - your program should expect an ***absolute path*** to a folder containing a set of json documents to parse.
    a. **Ignore any files in this folder that don't end in .json.**
    b. This folder will contain other folders like the original data set.
2. a search term - after indexing all documents located at the 2nd command line argument, you will print all of the document IDs that contain this search term.

Example:

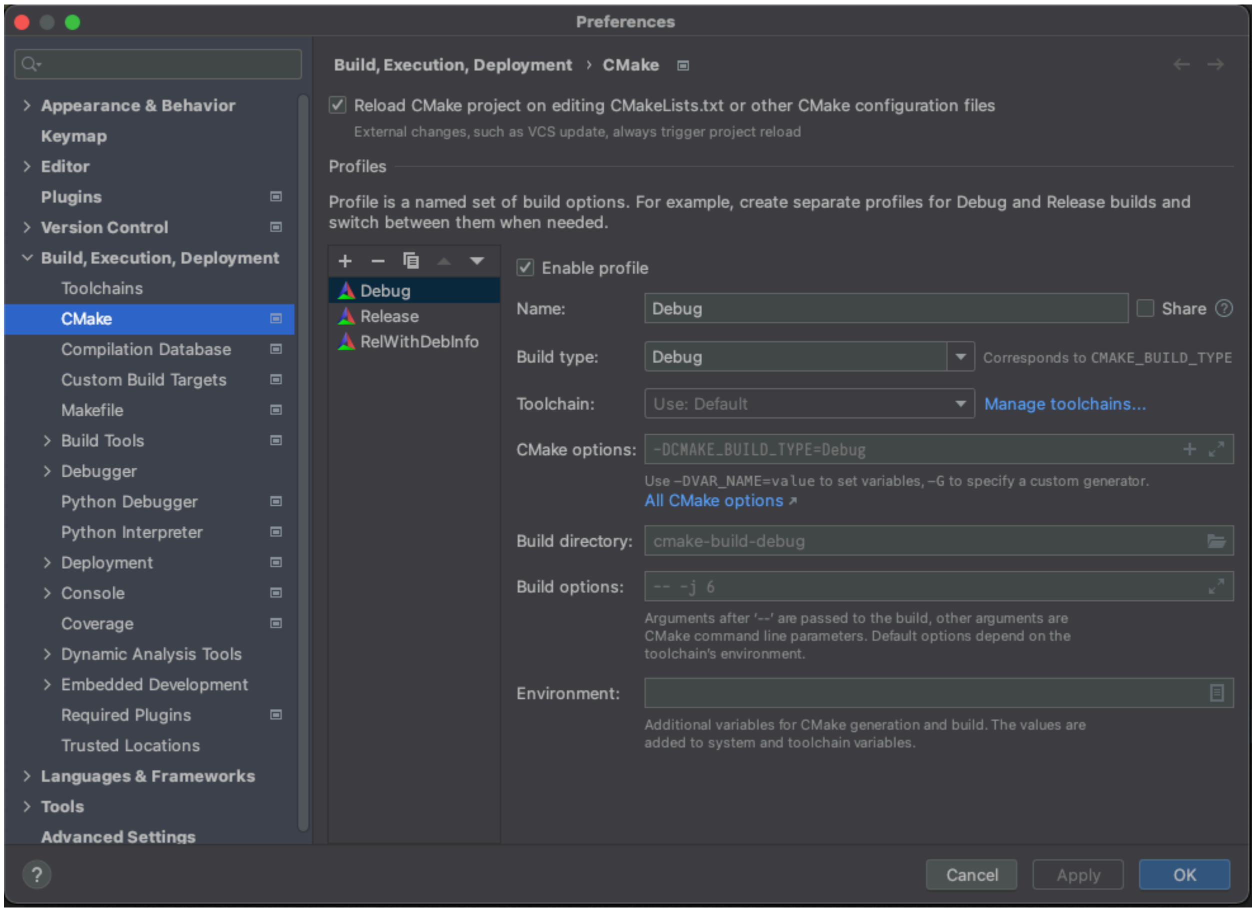
```
./a.out /home/mark/classes/2341/dataset01 pathogenesis
```

If your computer username contains a space, DO NOT change your username to accommodate the path above. Simply use finder or Windows Explorer to create a new folder outside of the User home directory folder. You might need admin privileges to do this.

Debug vs Release Mode

When you build your programs normally, the executables include extra info to allow you to use the debugger to step through your code. Also, no compiler optimization of your code is performed. These things make the execution slow.

You can also build your project in Release Mode, which removes any debugging symbols in the executable as well as cranks up the optimization of the compiler. You can add Release Build to your build configuration options by going to Clion Preferences > Build, Execution, Deployment > CMake. Then in the pane on the right side of the preferences dialog, click the + sign under Profiles to add a Release Profile.



☑ Reload CMake project on editing CMakeLists.txt or other CMake configuration files

External changes, such as VCS update, always trigger project reload

Profiles

Profile is a named set of build options. For example, create separate profiles for Debug and Release builds and switch between them when needed.

☑ Enable profile

Name: Debug ☐ Share

Build type: Debug Corresponds to CMAKE_BUILD_TYPE

Toolchain: Use: Default

CMake options: -DCMAKE_BUILD_TYPE=Debug

Use -DVAR_NAME=value to set variables, -G to specify a custom generator.

Build directory: cmake-build-debug

Build options: -- -j 6

Arguments after '--' are passed to the build, other arguments are CMake command line parameters. Default options depend on the toolchain's environment.

Environment:

Additional variables for CMake generation and build. The values are added to system and toolchain variables.