When developing on FDB it may be important to run a .cpp/.py/.java file using the client bindings built locally (for example you added a new function to the C API and want to test it).

Using a local version of the CAPI

- Set LD_LIBRARY_PATH to point to a directory containing libfdb_c.so using the following command: export LD_LIBRARY_PATH=/home/<user>/build/lib/
- 2. Compile your C++ file(s) using a command similar to: g++ -o <exe_name> -I ~/build/bindings/c/foundationdb/ -I ~/build/bindingtester/tests/c/ -std=c++17 -L ~/build/bindings/c/ -lfdb_c -lpthread a.cpp b.cpp
- When you now run ./<exe_name> it should reflect any changes you made to the C client API

Troubleshooting

If you're running into issues (i.e a new function you added is not showing up) try running ldd <path-to-binary> to check the library dependencies of a binary (to make sure it's loading libfdb_c correctly). # Using a local version of the Python API

- 1. export LD_LIBRARY_PATH=/home/<user>/build/lib/
- Given a python file you can simply run the following command: PYTHON-PATH=~/build/bindings/python/ python <file>.py

Using a local version of the Java API

- 1. export LD_LIBRARY_PATH=/home/<user>/build/lib/
- 2. First compile your file using the following command: javac -cp ".:/home/<user>/build/packages/fdb-java-7.0.0-PRERELEASE.jar" Example.java
- 3. Run the java file as follows: java -cp ".:/home/<user>/build/packages/fdb-java-7.0.0-PRERELEASE.jar" Example.java