# User

I am on Fedora Linux. I downloaded c++ DataFrame - a github project from:

https://github.com/hosseinmoein/DataFrame

While compiling, configuration was okay with: cmake -G Ninja .. -DCMAKE\_BUILD\_TYPE=Release -DCMAKE\_INSTALL\_PREFIX=/opt/cppdataframe -DHMDF\_BENCHMARKS=1 -DHMDF\_EXAMPLES=1 -DHMDF\_TESTING=1

But the next step cmake --build . resulted in errors (Partial output):

[5/45] Building CXX object CMakeFiles/DataFrame.dir/src/Utils/DateTime.cc.o
FAILED: CMakeFiles/DataFrame.dir/src/Utils/DateTime.cc.o
/usr/bin/c++ -DHMDF\_HAVE\_CLOCK\_GETTIME -I/home/admin/Downloads/DataFrame/include -O3 -DNDEBUG -std=gnu++23 -MD -MT CMakeFiles/DataFrame.dir/src/Utils/DateTime.cc.o -MF CMakeFiles/DataFrame.dir/src/Utils/DateTime.cc.o.d -o CMakeFiles/DataFrame.dir/src/Utils/DateTime.cc.o -c /home/admin/Downloads/DataFrame/src/Utils/DateTime.cc
In file included from /home/admin/Downloads/DataFrame/include/DataFrame/Utils/DateTime.h:33,
from /home/admin/Downloads/DataFrame/src/Utils/DateTime.cc:30:
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h: In member function ‘hmdf::VirtualString& hmdf::VirtualString::ncopy(const\_pointer, size\_type)’:
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h:42:61: error: ‘abort’ was not declared in this scope
42 | #define snprintf\_nowarn(...) (::snprintf(**VA\_ARGS**) < 0 ? abort() : (void)0)
| ^~~~~
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h:100:9: note: in expansion of macro ‘snprintf\_nowarn’
100 | snprintf\_nowarn(string\_, len, "%s", rhs);
| ^~~~~~~~~~~~~~~
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h:36:1: note: ‘abort’ is defined in header ‘<cstdlib>’; this is probably fixable by adding ‘#include <cstdlib>’
35 | #include <functional>
+++ |+#include <cstdlib>
36 |
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h: In member function ‘hmdf::FixedSizeString<S>& hmdf::FixedSizeString<S>::operator=(hmdf::VirtualString::const\_pointer)’:
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h:42:61: error: there are no arguments to ‘abort’ that depend on a template parameter, so a declaration of ‘abort’ must be available [-fpermissive]
42 | #define snprintf\_nowarn(...) (::snprintf(**VA\_ARGS**) < 0 ? abort() : (void)0)
| ^~~~~
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h:358:9: note: in expansion of macro ‘snprintf\_nowarn’
358 | snprintf\_nowarn(buffer\_, S, "%s", rhs);
| ^~~~~~~~~~~~~~~
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h:42:61: note: (if you use ‘-fpermissive’, G++ will accept your code, but allowing the use of an undeclared name is deprecated)
42 | #define snprintf\_nowarn(...) (::snprintf(**VA\_ARGS**) < 0 ? abort() : (void)0)
| ^~~~~
/home/admin/Downloads/DataFrame/include/DataFrame/Utils/FixedSizeString.h:358:9: note: in expansion of macro ‘snprintf\_nowarn’
358 | snprintf\_nowarn(buffer\_, S, "%s", rhs);
| ^~~~~~~~~~~~~~~

# Bing

The error message indicates that the abort function is not declared in the scope where it's being used. This function is defined in the <cstdlib> header file. The error message also suggests that adding #include <cstdlib> could probably fix the issue.

You can try adding #include <cstdlib> to the FixedSizeString.h file, just after the #include <functional> line. Here's how you can do it:

#include <functional>

#include <cstdlib>

After making this change, try to build the project again. If you still encounter issues, please let me know. I'm here to help! 😊

# User

Can I exclude from:

cmake -G Ninja .. -DCMAKE\_BUILD\_TYPE=Release -DCMAKE\_INSTALL\_PREFIX=/opt/cppdataframe -DHMDF\_BENCHMARKS=1 -DHMDF\_EXAMPLES=1 -DHMDF\_TESTING=1

DHMDF\_BENCHMARKS=1 -DHMDF\_EXAMPLES=1 -DHMDF\_TESTING=1 - these options? How they will affect performance or any other functionality?

# Bing

Yes, you can exclude the -DHMDF\_BENCHMARKS=1 -DHMDF\_EXAMPLES=1 -DHMDF\_TESTING=1 options from your cmake command. These options are used to control the building of benchmarks, examples, and tests respectively for the DataFrame project. Here's what each option does:

* -DHMDF\_BENCHMARKS=1: This option enables the building of benchmarks for the DataFrame project. Benchmarks are used to measure the performance of the code. If you exclude this option, the benchmarks will not be built.
* -DHMDF\_EXAMPLES=1: This option enables the building of examples for the DataFrame project. Examples are small programs that demonstrate how to use the DataFrame library. If you exclude this option, the examples will not be built.
* -DHMDF\_TESTING=1: This option enables the building of tests for the DataFrame project. Tests are used to verify that the code is working correctly. If you exclude this option, the tests will not be built.

Excluding these options will not affect the performance of the DataFrame library itself. They only control whether the benchmarks, examples, and tests are built. If you're just interested in using the DataFrame library and don't need the benchmarks, examples, or tests, you can exclude these options. Your cmake command would then look like this:

cmake -G Ninja .. -DCMAKE\_BUILD\_TYPE=Release -DCMAKE\_INSTALL\_PREFIX=/opt/cppdataframe

After running this command, you can build the project with cmake --build .. If you still encounter issues, please let me know. I'm here to help! 😊

# User

In my fedora even though I installed cmake using sudo dnf -y install cmake, when I run:

cmake -G Ninja .. -DCMAKE\_BUILD\_TYPE=Release -DCMAKE\_INSTALL\_PREFIX=/opt/cppdataframe

the message is:

bash: /home/admin/.local/bin/cmake: No such file or directory

# Bing

The error message suggests that the system is trying to find cmake in the /home/admin/.local/bin/ directory, but it's not there. This could be due to a problem with your PATH environment variable.

You can check where cmake is installed by using the which command:

which cmake

This command will print the path to the cmake executable. If cmake is installed correctly, you should see a path like /usr/bin/cmake.

If which cmake doesn't return anything or returns an error, it means that cmake is not found in the directories listed in your PATH environment variable. You can check the value of PATH with:

echo $PATH

If the directory containing cmake is not in PATH, you can add it. Suppose cmake is in /usr/bin, you can add it to PATH with:

export PATH=$PATH:/usr/bin

After updating PATH, try running the cmake command again. If you still encounter issues, please let me know. I'm here to help! 😊

# User

Although I could install at /opt/cppdataframe during compiling my following test code, I encountered errors in Fedora:

#include <DataFrame/DataFrame.h>
#include <random>
#include <fstream>

int main() {
// Set the optimum thread level for parallel processing
hmdf::ThreadGranularity::set\_thread\_level(4);
// hmdf::ThreadGranularity::set\_optimum\_thread\_level();
// Create a DataFrame object
hmdf::StdDataFrame<unsigned long> df;

// Create vectors to hold data

std::vector<double> vec1(10000000);

std::vector<int> vec2(10000000);

std::vector<std::string> vec3(10000000, "test");

// Create an index for the DataFrame

std::vector<unsigned long> index(10000000);

std::iota(index.begin(), index.end(), 0); // Fill with consecutive numbers

// Fill vec1 and vec2 with random numbers

std::random\_device rd;

std::mt19937 gen(rd());

std::uniform\_real\_distribution<> dis1(1.0, 2.0);

std::uniform\_int\_distribution<> dis2(1, 10);

for (auto &x : vec1) {

x = dis1(gen);

}

for (auto &x : vec2) {

x = dis2(gen);

}

// Load index into the DataFrame

df.load\_index(std::move(index));

// Load data into the DataFrame

df.load\_column("col1", std::move(vec1));

df.load\_column("col2", std::move(vec2));

df.load\_column("col3", std::move(vec3));

// Save the DataFrame to a CSV file

std::ofstream outfile("dataframe.csv");

df.write<std::ostream, int, double, std::string>(outfile);

return 0;

}

Errors:

[admin@fedora-rh-usb Downloads]$ g++ -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib testdataframe.cpp -o testdataframe
In file included from /usr/include/c++/14/x86\_64-redhat-linux/bits/c++config.h:2736,
from /usr/include/c++/14/limits:42,
from /opt/cppdataframe/include/DataFrame/Utils/AlignedAllocator.h:32,
from /opt/cppdataframe/include/DataFrame/Vectors/HeteroVector.h:32,
from /opt/cppdataframe/include/DataFrame/DataFrameTypes.h:32,
from /opt/cppdataframe/include/DataFrame/DataFrameStatsVisitors.h:32,
from /opt/cppdataframe/include/DataFrame/DataFrame.h:32,
from testdataframe.cpp:1:
/usr/include/c++/14/pstl/algorithm\_impl.h: In function ‘\_RandomAccessIterator \_\_pstl::\_\_internal::\_\_brick\_unique(\_RandomAccessIterator, \_RandomAccessIterator, \_BinaryPredicate, std::true\_type)’:
/usr/include/c++/14/pstl/algorithm\_impl.h:1219:5: note: ‘#pragma message: [Parallel STL message]: "Vectorized algorithm unimplemented, redirected to serial"’
1219 | \_PSTL\_PRAGMA\_MESSAGE("Vectorized algorithm unimplemented, redirected to serial");
| ^~~~~~~~~~~~~~~~~~~~
/usr/include/c++/14/pstl/algorithm\_impl.h: In function ‘\_RandomAccessIterator \_\_pstl::\_\_internal::\_\_brick\_partition(\_RandomAccessIterator, \_RandomAccessIterator, \_UnaryPredicate, std::true\_type)’:
/usr/include/c++/14/pstl/algorithm\_impl.h:1929:5: note: ‘#pragma message: [Parallel STL message]: "Vectorized algorithm unimplemented, redirected to serial"’
1929 | \_PSTL\_PRAGMA\_MESSAGE("Vectorized algorithm unimplemented, redirected to serial");
| ^~~~~~~~~~~~~~~~~~~~
/usr/include/c++/14/pstl/algorithm\_impl.h: In function ‘\_RandomAccessIterator \_\_pstl::\_\_internal::\_\_brick\_stable\_partition(\_RandomAccessIterator, \_RandomAccessIterator, \_UnaryPredicate, std::true\_type)’:
/usr/include/c++/14/pstl/algorithm\_impl.h:2029:5: note: ‘#pragma message: [Parallel STL message]: "Vectorized algorithm unimplemented, redirected to serial"’
2029 | \_PSTL\_PRAGMA\_MESSAGE("Vectorized algorithm unimplemented, redirected to serial");
| ^~~~~~~~~~~~~~~~~~~~
/usr/include/c++/14/pstl/algorithm\_impl.h: In function ‘\_RandomAccessIterator3 \_\_pstl::\_\_internal::\_\_brick\_merge(\_RandomAccessIterator1, \_RandomAccessIterator1, \_RandomAccessIterator2, \_RandomAccessIterator2, \_RandomAccessIterator3, \_Compare, std::true\_type)’:
/usr/include/c++/14/pstl/

# Bing

The error messages you're seeing are actually not errors, but rather informational messages from the Parallel STL library. The messages are indicating that certain vectorized (parallel) algorithms are not implemented and are being redirected to their serial (non-parallel) counterparts.

These messages are generated by #pragma message directives in the Parallel STL code. They don't indicate a problem with your code or with the DataFrame library. Your program should still compile and run correctly despite these messages.

However, if you're seeing these messages and your program is not compiling successfully, there may be other error messages that are indicating the actual problem. If that's the case, could you please provide the full output from the compiler? That would help me better understand what might be going wrong. 😊

# User

With g++ -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib testdataframe.cpp -o testdataframe

program compiled and ran okay - I agree - above messages are as you say STL type messages.

But using clang to compile still not success:

clang: error: linker command failed with exit code 1 (use -v to see invocation)
[admin@fedora-rh-usb Downloads]$ clang -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib testdataframe.cpp -o testdataframe -lstdc++ -pthread

In file included from testdataframe.cpp:1:
In file included from /opt/cppdataframe/include/DataFrame/DataFrame.h:32:
In file included from /opt/cppdataframe/include/DataFrame/DataFrameStatsVisitors.h:33:
In file included from /opt/cppdataframe/include/DataFrame/Internals/DataFrame\_standalone.tcc:33:
In file included from /opt/cppdataframe/include/DataFrame/Utils/Threads/ThreadGranularity.h:32:
In file included from /opt/cppdataframe/include/DataFrame/Utils/Threads/ThreadPool.h:186:
/opt/cppdataframe/include/DataFrame/Utils/Threads/ThreadPool.tcc:78:34: warning: absolute value function 'abs' given an argument of type 'size\_type' (aka 'long') but has parameter of type 'int' which may cause truncation of value [-Wabsolute-value]
78 | const size\_type shutys { ::abs(thr\_num) };
| ^
/opt/cppdataframe/include/DataFrame/Utils/Threads/ThreadPool.tcc:78:34: note: use function 'std::abs' instead
78 | const size\_type shutys { ::abs(thr\_num) };
| ^~~~~
| std::abs
1 warning generated.
/usr/bin/ld: /tmp/testdataframe-601902.o: undefined reference to symbol 'logl@@GLIBC\_2.2.5'
/usr/bin/ld: /usr/lib64/libm.so.6: error adding symbols: DSO missing from command line
clang: error: linker command failed with exit code 1 (use -v to see invocation)

Okay about warnings. But what about errors?

# Bing

The error message you're seeing is a linker error. It's complaining about an undefined reference to the symbol logl@@GLIBC\_2.2.5, which is a function from the math library (libm). This suggests that the math library is not being linked correctly.

In your command, you're using -lstdc++ -pthread to link the C++ standard library and the pthread library. However, you're not explicitly linking the math library. You can do this by adding -lm to your command.

Here's the modified command:

clang -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib testdataframe.cpp -o testdataframe -lstdc++ -pthread -lm

After running this command, your program should compile successfully. If you still encounter issues, please let me know. I'm here to help! 😊

# User

While using g++ to compile test code:

/usr/include/c++/14/pstl/algorithm\_impl.h: In function ‘\_RandomAccessIterator3 \_\_pstl::\_\_internal::\_\_brick\_set\_symmetric\_difference(\_RandomAccessIterator1, \_RandomAccessIterator1, \_RandomAccessIterator2, \_RandomAccessIterator2, \_RandomAccessIterator3, \_Compare, std::true\_type)’:
/usr/include/c++/14/pstl/algorithm\_impl.h:3390:5: note: ‘#pragma message: [Parallel STL message]: "Vectorized algorithm unimplemented, redirected to serial"’
3390 | \_PSTL\_PRAGMA\_MESSAGE("Vectorized algorithm unimplemented, redirected to serial");
| ^~~~~~~~~~~~~~~~~~~~

What does above indicate? It disables parallel processing? Requires some libraries?

# Bing

The message you're seeing is a note from the Parallel STL (Standard Template Library). It's indicating that a certain algorithm, in this case set\_symmetric\_difference, does not have a vectorized (parallel) implementation available, so it's being redirected to a serial (non-parallel) version.

Parallel STL is a part of the C++ Standard Library that provides parallel versions of many of the algorithms in the STL. These parallel algorithms can take advantage of multiple cores to run faster than their serial counterparts. However, not all algorithms have parallel versions, and in those cases, the Parallel STL falls back to using the serial version.

This message does not disable parallel processing for all algorithms, just for the set\_symmetric\_difference algorithm. Other algorithms that have parallel versions will still be able to run in parallel.

This message does not indicate that you're missing any libraries. It's just an informational message about how the Parallel STL is handling this particular algorithm. Your code should still compile and run correctly despite this message. If you're encountering any issues, please let me know! 😊

# User

But I did not see the message when I compiled in msys2 on Windows.

# Bing

The Parallel STL (Standard Template Library) is an extension of the C++ STL that provides parallel versions of many of the algorithms in the STL. The availability and behavior of these parallel algorithms can vary between different systems and compilers.

In your case, when you compiled your code on Fedora Linux with g++, you saw a message indicating that the set\_symmetric\_difference algorithm does not have a parallel implementation and was redirected to a serial one. However, when you compiled the same code on Windows with MSYS2, you did not see this message.

This could be due to a few reasons:

1. **Different STL Implementations**: The GNU Compiler Collection (GCC, which includes g++) and MSYS2 might be using different implementations of the STL, which could explain the difference in behavior.
2. **Compiler Flags**: Different compiler flags might have been used in the two environments, which could affect the behavior of the STL.
3. **Parallel STL Support**: The version of the STL used by MSYS2 might not include support for Parallel STL, or it might implement it differently.

In any case, these messages are informational and do not indicate a problem with your code. They are intended to inform you about the behavior of the STL and do not affect the correctness or performance of your program. If your program is working correctly, you can safely ignore these messages. 😊

# User

Another query is on Windows I compiled using:

g++ -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib -l DataFrame testdataframe.cpp -o testdataframe

On linux:

g++ -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib testdataframe.cpp -o testdataframe

Why? If I include -l DataFrame on Fedora, it complains and results in error.

# Bing

The -l option in the g++ command is used to link with a library. When you specify -l DataFrame, g++ looks for a library named libDataFrame.a or libDataFrame.so in the library paths, which are specified by -L options and some default paths.

In your Windows environment, it seems there is a libDataFrame.a or libDataFrame.so file in the /opt/cppdataframe/lib directory or one of the default library paths, so -l DataFrame works.

However, in your Fedora environment, g++ might not be finding libDataFrame.a or libDataFrame.so in the specified library paths, hence the error when you include -l DataFrame.

It's also possible that the DataFrame library is built into the object files on Fedora, so you don't need to link against a separate DataFrame library, which is why the command works without -l DataFrame.

To confirm this, you could check if there's a libDataFrame.a or libDataFrame.so file in the /opt/cppdataframe/lib directory on your Fedora system. If not, that's likely why -l DataFrame results in an error. 😊

# User

I found that /opt/cppdataframe/lib64 directory has file named libDataFrame.a

# Bing

Great! You've found the libDataFrame.a file in the /opt/cppdataframe/lib64 directory. This is the static library file for DataFrame.

Now, when you're compiling your program, you need to tell g++ to look in the /opt/cppdataframe/lib64 directory for library files. You can do this by using the -L option followed by the directory path. Also, you need to link against the DataFrame library using -l DataFrame.

Here's how you can modify your command:

g++ -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib64 -l DataFrame testdataframe.cpp -o testdataframe

This command tells g++ to:

* Use C++23 standard (-std=c++23)
* Look for include files in /opt/cppdataframe/include (-I /opt/cppdataframe/include)
* Look for library files in /opt/cppdataframe/lib64 (-L /opt/cppdataframe/lib64)
* Link against the DataFrame library (-l DataFrame)
* Compile testdataframe.cpp
* Output the executable to testdataframe

After running this command, your program should compile successfully. If you still encounter issues, please let me know. I'm here to help! 😊

# User

With:

g++ -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib -l DataFrame testdataframe.cpp -o testdataframe

Compilation error:

/usr/bin/ld: cannot find -lDataFrame: No such file or directory
collect2: error: ld returned 1 exit status

Without -l DataFrame compiles successfully

# Bing

I apologize for the confusion. The error message indicates that the linker cannot find the DataFrame library in the /opt/cppdataframe/lib directory. However, you mentioned earlier that the libDataFrame.a file is located in the /opt/cppdataframe/lib64 directory.

So, you should use -L /opt/cppdataframe/lib64 instead of -L /opt/cppdataframe/lib in your command to tell the linker to look in the correct directory. Here's the corrected command:

g++ -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib64 -l DataFrame testdataframe.cpp -o testdataframe

If your program compiles successfully without -l DataFrame, it's possible that the DataFrame library is header-only or included in the source files, so you don't need to link against a separate DataFrame library. If that's the case, you can continue to compile without -l DataFrame. 😊