# User

See below my code of c++:

#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;

}

Compiled using:

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

Compilation without error

Segementation fault (runtime)

gdb output:

(gdb) run
Starting program: D:\Programs\Msys2\home\juhi123\Downloads\testdataframeclang.exe
[New Thread 14636.0x2484]
[New Thread 14636.0x2190]
[New Thread 14636.0x708]
[New Thread 14636.0x14e0]
[New Thread 14636.0x4870]
[New Thread 14636.0x2d5c]
[New Thread 14636.0x2218]

Thread 6 received signal SIGSEGV, Segmentation fault.
[Switching to Thread 14636.0x4870]
0x00007ff7cf6fb725 in hmdf::ThreadPool::thread\_routine\_(long) ()

(gdb) backtrace
#0 0x00007ff7cf6fb725 in hmdf::ThreadPool::thread\_routine\_(long) ()
#1 0x00007ff7cf724632 in bool std::\_\_invoke\_impl<bool, bool (hmdf::ThreadPool::*)(long) noexcept, hmdf::ThreadPool*, long>(std::\_\_invoke\_memfun\_deref, bool (hmdf::ThreadPool::*&&)(long) noexcept, hmdf::ThreadPool*&&, long&&) ()
#2 0x00007ff7cf72804b in std::\_\_invoke\_result<bool (hmdf::ThreadPool::*)(long) noexcept, hmdf::ThreadPool*, long>::type std::\_\_invoke<bool (hmdf::ThreadPool::*)(long) noexcept, hmdf::ThreadPool*, long>(bool (hmdf::ThreadPool::*&&)(long) noexcept, hmdf::ThreadPool*&&, long&&) ()
#3 0x00007ff7cf7154dc in bool std::thread::\_Invoker<std::tuple<bool (hmdf::ThreadPool::*)(long) noexcept, hmdf::ThreadPool*, long> >::\_M\_invoke<0ull, 1ull, 2ull>(std::\_Index\_tuple<0ull, 1ull, 2ull>)
()
#4 0x00007ff7cf715587 in std::thread::\_Invoker<std::tuple<bool (hmdf::ThreadPool::*)(long) noexcept, hmdf::ThreadPool*, long> >::operator()() ()
#5 0x00007ff7cf715277 in std::thread::\_State\_impl<std::thread::\_Invoker<std::tuple<bool (hmdf::ThreadPool::*)(long) noexcept, hmdf::ThreadPool*, long> > >::\_M\_run() ()
#6 0x00007ffd1527d80f in ?? () from D:\Programs\mingw64\bin\libstdc++-6.dll
#7 0x00007ffdac9182f5 in ?? () from D:\Programs\mingw64\bin\libwinpthread-1.dll
#8 0x00007ffdde889363 in ucrtbase!\_recalloc () from C:\Windows\System32\ucrtbase.dll
#9 0x00007ffddf55257d in KERNEL32!BaseThreadInitThunk () from C:\Windows\System32\kernel32.dll
#10 0x00007ffde0c8aa58 in ntdll!RtlUserThreadStart () from C:\Windows\SYSTEM32\ntdll.dll
#11 0x0000000000000000 in ?? ()

# Bing

The segmentation fault is occurring in the hmdf::ThreadPool::thread\_routine\_(long) function. This function is part of the DataFrame library you're using, and it's related to the multithreading functionality of the library.

The error is likely due to a problem with how threads are being managed or used in your program. It could be a race condition, a problem with how resources are being shared between threads, or an issue with the thread pool size.

Here are a few things you could try to debug this issue:

1. **Check the thread level**: You've set the thread level to 4 using hmdf::ThreadGranularity::set\_thread\_level(4);. Try adjusting this value to see if the error persists. You could also uncomment the line hmdf::ThreadGranularity::set\_optimum\_thread\_level(); to let the library determine the optimal thread level.
2. **Inspect the data**: Ensure that the data you're loading into the DataFrame is valid and correctly formatted. If there's an issue with the data, it could potentially cause problems when the DataFrame operations are parallelized.
3. **Update the library**: If you're not using the latest version of the DataFrame library, consider updating it. The issue might be a bug that has been fixed in a more recent version.
4. **Simplify the code**: Try simplifying your code to the bare minimum that reproduces the error. This could make the problem easier to understand and debug.

Remember, multithreaded programming can be tricky due to the complexity of coordinating multiple threads. If you're new to it, I recommend reading up on the basics of multithreading and the specific multithreading features of the DataFrame library. Good luck! 👍

# User

If I comment out both:

hmdf::ThreadGranularity::set\_thread\_level(4); and
hmdf::ThreadGranularity::set\_optimum\_thread\_level();

then I can compile and run using clang command without errors. Interestingly, When I use following command with this same code without commenting as above):

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

No errors in compilation and no segmentation fault.

Why? Also I tried to compile in Fedora. Both clang and g++ compilation ran with parallel threading. What is wrong with clang on windows?

# Bing

The difference in behavior between g++ and clang, and between different operating systems, could be due to a variety of factors. Here are a few possibilities:

1. **Differences in the compilers**: g++ and clang are different compilers and may handle certain aspects of code compilation differently. This could potentially lead to differences in the compiled program's behavior.
2. **Differences in the standard libraries**: Each compiler comes with its own version of the C++ standard library. These libraries might have different implementations for certain features, which could lead to differences in behavior.
3. **Differences in the operating systems**: Different operating systems have different ways of handling threads and memory management. This could potentially affect the behavior of multithreaded programs.
4. **Differences in the DataFrame library**: The DataFrame library you're using might have different code paths for different compilers or operating systems. This could potentially lead to differences in behavior.

As for why commenting out the lines related to thread level resolves the issue, it's possible that the default thread level used by the DataFrame library is more suitable for your program or your system's configuration.

To get to the bottom of this, you might want to reach out to the maintainers of the DataFrame library or ask on a forum like Stack Overflow, providing all the details you've mentioned here. They might be able to provide more insight into why you're seeing this behavior. Good luck! 👍

# User

I could see that in linux when I compiled dataframe library from github, it created /opt/lib64 while in windows it created /opt/lib. Could 64 bit issue may be there?

# Bing

Yes, the difference between /opt/lib64 and /opt/lib could potentially indicate a difference in architecture (64-bit vs 32-bit).

In many Linux distributions, lib64 is used for 64-bit libraries, while lib is used for 32-bit libraries. However, this convention may not hold true for all systems. On some systems, lib can hold the libraries for the native architecture, and others (like lib32 or libx32) are used for the non-native ones.

In your case, if the DataFrame library was compiled for a 64-bit architecture on Linux (hence the lib64), and you're trying to use it in a 32-bit environment on Windows (hence the lib), that could indeed cause issues.

You might want to ensure that you're consistently using the same architecture (either 32-bit or 64-bit) across all your environments. If your Windows system is 64-bit, you should be using 64-bit libraries. If it's 32-bit, you should be using 32-bit libraries.

Please check the architecture of your Windows system and ensure that the correct version of the DataFrame library is being used. If the problem persists, you might want to reach out to the maintainers of the DataFrame library or ask on a forum like Stack Overflow, providing all the details you've mentioned here. They might be able to provide more insight into why you're seeing this behavior. Good luck! 👍

# User

How to use cmake forcefully build using 64 bit libraries?

# Bing

To force CMake to build for a 64-bit architecture, you can use a toolchain file or set the CMAKE\_GENERATOR\_PLATFORM variable. Here's how you can do it:

**Method 1: Using a Toolchain File**

Create a toolchain file, say toolchain-x64.cmake, with the following content:

set(CMAKE\_SYSTEM\_NAME Windows)

set(CMAKE\_SYSTEM\_PROCESSOR AMD64)

Then, when running CMake, specify the toolchain file:

cmake -DCMAKE\_TOOLCHAIN\_FILE=toolchain-x64.cmake ..

**Method 2: Setting the Generator Platform**

When running CMake, you can specify the generator platform:

cmake -G "Visual Studio 16 2019" -A x64 ..

In this command, -G specifies the generator to use and -A specifies the platform. Replace "Visual Studio 16 2019" with the generator you're using.

Please note that these methods are specific to the system and the compiler you are using. The exact commands might differ based on your setup. If you're not using Visual Studio or if you're on a different system, you'll need to adjust the commands accordingly.

Also, ensure that the libraries you're linking against are available in 64-bit versions. If they're not, you'll need to obtain or build 64-bit versions of those libraries.

Remember to clean your build directory before re-running CMake with these new settings. Good luck! 👍

# User

Can I use:

cmake -G Ninja .. -A64 -DCMAKE\_BUILD\_TYPE=Release -DCMAKE\_INSTALL\_PREFIX=/opt/cppdataframe

# Bing

The -A option is not available with Ninja. The -A option is used with Visual Studio to set the target architecture. Ninja does not have a similar option.

However, Ninja will use the architecture of the environment in which it's run. So if you're running a 64-bit shell, Ninja will build for a 64-bit target.

Here's how you can use cmake with Ninja:

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

This command will generate the build files for Ninja. You can then build the project by running ninja in the build directory.

Remember to ensure that your environment matches the target architecture (64-bit in this case). If you're using a different shell or system, you might need to adjust the commands accordingly. Good luck! 👍

# User

How to force msys2 shell use 64 bit?

# Bing

MSYS2 shell uses the architecture of the environment in which it's run. If you're running a 64-bit shell, MSYS2 will build for a 64-bit target.

If you want to ensure that MSYS2 is using the 64-bit architecture, you can use the "MSYS2 MinGW 64-bit" launcher in the "MSYS2 64bit" directory[**1**](https://superuser.com/questions/961699/change-default-shell-on-msys2).

If you want to change the default shell, you can modify the msys2\_shell.cmd command in the target of the launcher shortcut. For example, if you want to use zsh as your default shell, you can change the target to:

C:\msys64\msys2\_shell.cmd -mingw64 -shell zsh

This command will launch the MSYS2 shell using the zsh shell[**1**](https://superuser.com/questions/961699/change-default-shell-on-msys2).

Please ensure that you have the correct shell installed and that the path to the shell is correct. If you're not sure, you can check the available shells by running chsh -l in the MSYS2 shell.

Remember to restart your MSYS2 shell or your computer for the changes to take effect. Good luck! 👍

Source(s)

1. [windows - Change default shell on MSYS2 - Super User](https://superuser.com/questions/961699/change-default-shell-on-msys2)

2. [Terminals - MSYS2](https://www.msys2.org/docs/terminals/)

3. [windows - Executing a script in MSYS2/MinGW - Stack Overflow](https://stackoverflow.com/questions/47438779/executing-a-script-in-msys2-mingw)

4. <https://github.com/msys2/msys2-launcher>

5. <https://mintty.github.io>

6. <https://docs.microsoft.com/en-us/windows/terminal/customize-settings/profile-settings>

# User

See I could not avoid that segmentation fault using clang packaged with gcc by Brecht Sanders downloaded from github so I decided to fall back on default tool-chain of clang packaged for Msys2 and installed in shell using pacman -S.

When I compiled above c++ code, it worked as expected but certain warnings as below were generated:

$ clang -std=c++23 -I /opt/cppdataframe/include -L /opt/cppdataframe/lib testdataframe.cpp -o testdataframeclang -lstdc++ -pthread -lm
In file included from testdataframe.cpp:1:
In file included from D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/DataFrame.h:4251:
D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/Internals/DataFrame\_set.tcc:695:18: warning:
format specifies type 'unsigned long' but the argument has type 'size\_type'
(aka 'unsigned long long') [-Wformat]
693 | "data size of %lu is larger than index size of %lu",
| ~~~
| %zu
694 | #endif // \_MSC\_VER
695 | data\_s, idx\_s);
| ^~~~~~
testdataframe.cpp:39:8: note: in instantiation of function template specialization
'hmdf::DataFrame<unsigned long, hmdf::HeteroVector<>>::load\_column<double>' requested here
39 | df.load\_column("col1", std::move(vec1));
| ^
In file included from testdataframe.cpp:1:
In file included from D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/DataFrame.h:4251:
D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/Internals/DataFrame\_set.tcc:695:26: warning:
format specifies type 'unsigned long' but the argument has type 'size\_type'
(aka 'unsigned long long') [-Wformat]
693 | "data size of %lu is larger than index size of %lu",
| ~~~
| %zu
694 | #endif // \_MSC\_VER
695 | data\_s, idx\_s);
| ^~~~~
D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/Internals/DataFrame\_set.tcc:695:18: warning:
format specifies type 'unsigned long' but the argument has type 'size\_type'
(aka 'unsigned long long') [-Wformat]
693 | "data size of %lu is larger than index size of %lu",
| ~~~
| %zu
694 | #endif // \_MSC\_VER
695 | data\_s, idx\_s);
| ^~~~~~
testdataframe.cpp:40:8: note: in instantiation of function template specialization
'hmdf::DataFrame<unsigned long, hmdf::HeteroVector<>>::load\_column<int>' requested here
40 | df.load\_column("col2", std::move(vec2));
| ^
In file included from testdataframe.cpp:1:
In file included from D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/DataFrame.h:4251:
D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/Internals/DataFrame\_set.tcc:695:26: warning:
format specifies type 'unsigned long' but the argument has type 'size\_type'
(aka 'unsigned long long') [-Wformat]
693 | "data size of %lu is larger than index size of %lu",
| ~~~
| %zu
694 | #endif // \_MSC\_VER
695 | data\_s, idx\_s);
| ^~~~~
D:/Programs/Msys2/opt/cppdataframe/include/DataFrame/Internals/DataFrame\_set.tcc:695:18: warning:
format specifies type 'unsigned long' but the argument has type 'size\_type'
(aka 'unsigned long long') [-Wformat]
693 | "data size of %lu is larger than index size of %lu",
| ~~~
| %zu
694 | #endif // \_MSC\_VER

# Bing

The warnings you're seeing are due to a mismatch between the format specifier used in the printf-style function and the type of the argument provided. In your case, the format specifier %lu is used for unsigned long, but the argument provided is of type size\_type (which is unsigned long long).

To fix these warnings, you can change the format specifier to %zu, which is the correct format specifier for size\_t and unsigned long long on most platforms.

Here's how you can do it:

"Data size of %zu is larger than index size of %zu", data\_s, idx\_s

This change should be made in the DataFrame library's source code, specifically in the DataFrame\_set.tcc file. After making these changes, you should recompile the DataFrame library and then recompile your program.

Please note that modifying a library's source code can have unintended side effects, and it's generally recommended to report the issue to the library's maintainers so they can provide an official fix. If you're not comfortable making these changes, you might want to reach out to the DataFrame library's maintainers for assistance. Good luck! 👍

# User

But even with warnings code compiled and ran okay - So no issues. Should I ignore warnings or they have some impact on content of output csv file?

# Bing

Warnings in C++ are not errors and do not prevent your code from compiling or running. However, they indicate potential issues in your code that could lead to bugs or unexpected behavior.

In your case, the warnings are about a format specifier in a printf-style function that doesn't match the type of the argument provided. This mismatch doesn't affect the logic of your program or the content of the output CSV file. The warnings are related to how the program reports the sizes of the data and index, not how it processes or outputs the data.

However, it's generally a good practice to address warnings because they can sometimes indicate serious issues. Ignoring warnings might hide potential problems that could become harder to debug in the future.

If the warnings are coming from a library you're using (as in this case), and you're sure they don't affect your program, you can choose to ignore them. But it might be worth reporting them to the library's maintainers. They might be able to provide a fix or update the library to remove the warnings. 👍

# User

Where in DataFrame\_set.tcc I have to make following changes? In the beginning? Where is the location of that file?

# Bing

The DataFrame\_set.tcc file is located in the include/DataFrame/Internals directory of the DataFrame library[**2**](https://stackoverflow.com/questions/35384358/how-to-open-my-files-in-data-folder-with-pandas-using-relative-path).

The changes need to be made in the warning lines mentioned in your compiler output. Specifically, you need to replace %lu with %zu in the following line of code:

"data size of %lu is larger than index size of %lu"

This line should be changed to:

"data size of %zu is larger than index size of %zu"

This line is located in the DataFrame\_set.tcc file, inside the load\_column function. The exact line number may vary depending on the version of the DataFrame library you're using.

After making these changes, you should recompile the DataFrame library and then recompile your program. If you're not comfortable making these changes, you might want to reach out to the DataFrame library's maintainers for assistance. Good luck! 👍

Source(s)

1. [GitHub](https://github.com/hosseinmoein/DataFrame/blob/master/include/DataFrame/Internals/DataFrame_set.tcc)

2. [How to open my files in data\_folder with pandas using relative path?](https://stackoverflow.com/questions/35384358/how-to-open-my-files-in-data-folder-with-pandas-using-relative-path)

3. [pandas.DataFrame.loc — pandas 2.2.0 documentation](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html)

4. [GitHub: Let’s build from here · GitHub](https://github.com/hosseinmoein/DataFrame/blob/master/include/DataFrame/Internals/DataFrame.tcc)

5. <https://avatars.githubusercontent.com/u/31854960?v=4>

# User

But I find following type of logic in that file:

#ifdef \_MSC\_VER
"data size of %zu is larger than index size of %zu",
#else
"data size of %lu is larger than index size of %lu",
#endif // \_MSC\_VER

Is it still correct to follow your above recommendation?

# Bing

The code you've posted is using conditional compilation to check if the Microsoft Visual C++ compiler is being used. If it is, it uses %zu as the format specifier, otherwise it uses %lu.

The warnings you're seeing are coming from the else part of this conditional compilation, which means the Microsoft Visual C++ compiler is not being used.

Given this, it seems that the library is already trying to handle the difference in format specifiers between different compilers, but it might not be doing so correctly for your specific setup.

You could try replacing %lu with %zu in the else part of the conditional compilation and see if that resolves the warnings. However, please note that this might not be a universal solution and could potentially cause issues on other systems or with other compilers.

As always, it's a good idea to report these warnings to the library's maintainers. They might be able to provide a more robust solution that works across all systems and compilers. Good luck! 👍