# User

I want to build gcc with UCRT option on windows. Can you tell me the procedure?

I have mingw64 installation at present which has gcc 13.2. I want to build latest from github. My present configuration (when I gave command gcc -v in powershell command prompt, following was the output).

I want to build in exactly same way but find that output has something like winlibs64ucrt (so it seems ucrt has been built proprietarily by winlibs.com. See below my output of gcc -v command:

PS D:\Downloads> gcc -v

Using built-in specs.
COLLECT\_GCC=d:\Programs\mingw64\bin\gcc.exe
COLLECT\_LTO\_WRAPPER=D:/Programs/mingw64/bin/../libexec/gcc/x86\_64-w64-mingw32/13.2.0/lto-wrapper.exe
OFFLOAD\_TARGET\_NAMES=nvptx-none
Target: x86\_64-w64-mingw32
Configured with: ../configure --prefix=/R/winlibs64ucrt\_stage/inst\_gcc-13.2.0/share/gcc --build=x86\_64-w64-mingw32 --host=x86\_64-w64-mingw32 --enable-offload-targets=nvptx-none --with-pkgversion='MinGW-W64 x86\_64-ucrt-posix-seh, built by Brecht Sanders, r4' --with-tune=generic --enable-checking=release --enable-threads=posix --disable-sjlj-exceptions --disable-libunwind-exceptions --disable-serial-configure --disable-bootstrap --enable-host-shared --enable-plugin --disable-default-ssp --disable-rpath --disable-libstdcxx-debug --disable-version-specific-runtime-libs --with-stabs --disable-symvers --enable-languages=c,c++,fortran,lto,objc,obj-c++ --disable-gold --disable-nls --disable-stage1-checking --disable-win32-registry --disable-multilib --enable-ld --enable-libquadmath --enable-libada --enable-libssp --enable-libstdcxx --enable-lto --enable-fully-dynamic-string --enable-libgomp --enable-graphite --enable-mingw-wildcard --enable-libstdcxx-time --enable-libstdcxx-pch --with-mpc=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-mpfr=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-gmp=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-isl=/d/Prog/winlibs64ucrt\_stage/custombuilt --disable-libstdcxx-backtrace --enable-install-libiberty --enable-\_\_cxa\_atexit --without-included-gettext --with-diagnostics-color=auto --enable-clocale=generic --with-libiconv --with-system-zlib --with-build-sysroot=/R/winlibs64ucrt\_stage/gcc-13.2.0/build\_mingw/mingw-w64 CFLAGS='-I/d/Prog/winlibs64ucrt\_stage/custombuilt/include/libdl-win32 -march=nocona -msahf -mtune=generic -O2' CXXFLAGS='-Wno-int-conversion -march=nocona -msahf -mtune=generic -O2' LDFLAGS='-pthread -Wl,--no-insert-timestamp -Wl,--dynamicbase -Wl,--high-entropy-va -Wl,--nxcompat -Wl,--tsaware' LD=/d/Prog/winlibs64ucrt\_stage/custombuilt/share/binutils/bin/ld.exe
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 13.2.0 (MinGW-W64 x86\_64-ucrt-posix-seh, built by Brecht Sanders, r4)

# Bing

Sure, I can guide you through the process of building GCC with the UCRT option on Windows. Here are the steps:

1. **Install the Universal CRT SDK**: Run the Visual Studio Installer, select the "Modify" button, go to the "Individual Components" tab, scroll down to "Compilers, build tools and runtimes", tick "Windows Universal CRT SDK", and install[**3**](https://learn.microsoft.com/en-us/cpp/porting/upgrade-your-code-to-the-universal-crt?view=msvc-170).
2. **Change the way MinGW GCC operates to link against UCRT instead of MSVCRT**[**1**](https://stackoverflow.com/questions/38290169/cannot-find-corecrt-h-universalcrt-includepath-is-wrong):
   * Dump the specs file: gcc -dumpspecs > $(MINGW\_ROOT)/lib/gcc/i686-w64-mingw32/$(GCC\_VERSION)/specs
   * Add -D\_UCRT to cpp and cc1plus sections of the specs file. This will prevent undefined references linking errors for the scanf functions family.
   * Replace -lmsvcrt with -lucrt in the libgcc section of the specs file.

Please replace $(MINGW\_ROOT) with the location of where you have MinGW and $(GCC\_VERSION) with your GCC version.

1. **Clone the GCC repository and build GCC**:

# Clone the GCC repository

git clone https://github.com/gcc-mirror/gcc.git

# Navigate to the GCC directory

cd gcc

# Create a build directory and navigate to it

mkdir build

cd build

# Configure the build

../configure --prefix=/R/winlibs64ucrt\_stage/inst\_gcc-13.2.0/share/gcc --build=x86\_64-w64-mingw32 --host=x86\_64-w64-mingw32 --enable-offload-targets=nvptx-none --with-pkgversion='MinGW-W64 x86\_64-ucrt-posix-seh, built by Brecht Sanders, r4' --with-tune=generic --enable-checking=release --enable-threads=posix --disable-sjlj-exceptions --disable-libunwind-exceptions --disable-serial-configure --disable-bootstrap --enable-host-shared --enable-plugin --disable-default-ssp --disable-rpath --disable-libstdcxx-debug --disable-version-specific-runtime-libs --with-stabs --disable-symvers --enable-languages=c,c++,fortran,lto,objc,obj-c++ --disable-gold --disable-nls --disable-stage1-checking --disable-win32-registry --disable-multilib --enable-ld --enable-libquadmath --enable-libada --enable-libssp --enable-libstdcxx --enable-lto --enable-fully-dynamic-string --enable-libgomp --enable-graphite --enable-mingw-wildcard --enable-libstdcxx-time --enable-libstdcxx-pch --with-mpc=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-mpfr=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-gmp=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-isl=/d/Prog/winlibs64ucrt\_stage/custombuilt --disable-libstdcxx-backtrace --enable-install-libiberty --enable-\_\_cxa\_atexit --without-included-gettext --with-diagnostics-color=auto --enable-clocale=generic --with-libiconv --with-system-zlib --with-build-sysroot=/R/winlibs64ucrt\_stage/gcc-13.2.0/build\_mingw/mingw-w64 CFLAGS='-I/d/Prog/winlibs64ucrt\_stage/custombuilt/include/libdl-win32 -march=nocona -msahf -mtune=generic -O2' CXXFLAGS='-Wno-int-conversion -march=nocona -msahf -mtune=generic -O2' LDFLAGS='-pthread -Wl,--no-insert-timestamp -Wl,--dynamicbase -Wl,--high-entropy-va -Wl,--nxcompat -Wl,--tsaware' LD=/d/Prog/winlibs64ucrt\_stage/custombuilt/share/binutils/bin/ld.exe

# Build GCC

make -j$(nproc)

# Install GCC

make install

Please replace the paths and options in the configure command with the ones that match your system configuration and requirements.

Please note that building GCC from source can take a significant amount of time and resources. Make sure your system meets the necessary requirements before proceeding.

Let me know if you need further assistance!

Source(s)

1. [Cannot find corecrt.h: $(UniversalCRT\_IncludePath) is wrong](https://stackoverflow.com/questions/38290169/cannot-find-corecrt-h-universalcrt-includepath-is-wrong)

2. [How do I build against the UCRT with mingw-w64?](https://stackoverflow.com/questions/57528555/how-do-i-build-against-the-ucrt-with-mingw-w64)

3. [Upgrade your code to the Universal CRT | Microsoft Learn](https://learn.microsoft.com/en-us/cpp/porting/upgrade-your-code-to-the-universal-crt?view=msvc-170)

4. [gcc - Configuring Mingw64 for 64bit - Stack Overflow](https://stackoverflow.com/questions/21302595/configuring-mingw64-for-64bit)

5. [github.com](https://github.com/cdecl/cdecl.github.io/tree/92f44c025fc83b494c9b98c860322f0f76f30697/_posts%2F2021-08-27-gcc-install-mac.md)

# User

But I think Visual Studio (VS) installation requies admin rights. Is it not possible to build that step (Universal CRT SDK) without VS?

# Bing

Yes, you're correct. Installing Visual Studio does require admin rights. However, there are a few alternatives you can consider:

1. **Visual C++ Build Tools**: You can install just the C++ Build Tools, without a full Visual Studio IDE installation[**1**](https://stackoverflow.com/questions/22290501/can-i-download-the-visual-c-command-line-compiler-without-visual-studio). From the Visual Studio Downloads page, scroll down until you see "Tools for Visual Studio" under the "All downloads" section and select the download for "Build Tools for Visual Studio". This should include the Universal CRT SDK.
2. **Manual Installation**: If you're up for some manual installation, you can extract individual components from the Visual Studio bundle and put them all in a more organized directory tree[**1**](https://stackoverflow.com/questions/22290501/can-i-download-the-visual-c-command-line-compiler-without-visual-studio). For example, you can find the bare minimum needed for using the current MSVC compilers in a x86 environment.
3. **Universal CRT as a part of Windows 10**: The Universal CRT is a component of the Windows operating system. It's included as part of Windows 10, starting with the January Technical Preview, and it is available for older versions of the operating system via Windows Update[**2**](https://devblogs.microsoft.com/cppblog/introducing-the-universal-crt/)[**3**](https://learn.microsoft.com/en-us/cpp/windows/universal-crt-deployment?view=msvc-170).

Please note that these alternatives might still require admin rights for certain steps. If you don't have admin rights on your machine, you might need to ask your system administrator for assistance. Let me know if you need further help!

Source(s)

1. [Can I download the Visual C++ Command Line Compiler without Visual Studio?](https://stackoverflow.com/questions/22290501/can-i-download-the-visual-c-command-line-compiler-without-visual-studio)

2. [Introducing the Universal CRT - C++ Team Blog](https://devblogs.microsoft.com/cppblog/introducing-the-universal-crt/)

3. [Universal CRT deployment | Microsoft Learn](https://learn.microsoft.com/en-us/cpp/windows/universal-crt-deployment?view=msvc-170)

# User

So I have Windows 11. UCRT must have been built into it by default. Can you modify and give me steps above for GCC building considering this information?

# Bing

Sure, if you're using Windows 11, the Universal CRT should indeed be built into it by default. Here are the modified steps to build GCC with the UCRT option:

1. **Change the way MinGW GCC operates to link against UCRT instead of MSVCRT**:
   * Dump the specs file: gcc -dumpspecs > $(MINGW\_ROOT)/lib/gcc/i686-w64-mingw32/$(GCC\_VERSION)/specs
   * Add -D\_UCRT to cpp and cc1plus sections of the specs file. This will prevent undefined references linking errors for the scanf functions family.
   * Replace -lmsvcrt with -lucrt in the libgcc section of the specs file.

Please replace $(MINGW\_ROOT) with the location of where you have MinGW and $(GCC\_VERSION) with your GCC version.

1. **Clone the GCC repository and build GCC**:

# Clone the GCC repository

git clone https://github.com/gcc-mirror/gcc.git

# Navigate to the GCC directory

cd gcc

# Create a build directory and navigate to it

mkdir build

cd build

# Configure the build

../configure --prefix=/R/winlibs64ucrt\_stage/inst\_gcc-13.2.0/share/gcc --build=x86\_64-w64-mingw32 --host=x86\_64-w64-mingw32 --enable-offload-targets=nvptx-none --with-pkgversion='MinGW-W64 x86\_64-ucrt-posix-seh, built by Brecht Sanders, r4' --with-tune=generic --enable-checking=release --enable-threads=posix --disable-sjlj-exceptions --disable-libunwind-exceptions --disable-serial-configure --disable-bootstrap --enable-host-shared --enable-plugin --disable-default-ssp --disable-rpath --disable-libstdcxx-debug --disable-version-specific-runtime-libs --with-stabs --disable-symvers --enable-languages=c,c++,fortran,lto,objc,obj-c++ --disable-gold --disable-nls --disable-stage1-checking --disable-win32-registry --disable-multilib --enable-ld --enable-libquadmath --enable-libada --enable-libssp --enable-libstdcxx --enable-lto --enable-fully-dynamic-string --enable-libgomp --enable-graphite --enable-mingw-wildcard --enable-libstdcxx-time --enable-libstdcxx-pch --with-mpc=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-mpfr=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-gmp=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-isl=/d/Prog/winlibs64ucrt\_stage/custombuilt --disable-libstdcxx-backtrace --enable-install-libiberty --enable-\_\_cxa\_atexit --without-included-gettext --with-diagnostics-color=auto --enable-clocale=generic --with-libiconv --with-system-zlib --with-build-sysroot=/R/winlibs64ucrt\_stage/gcc-13.2.0/build\_mingw/mingw-w64 CFLAGS='-I/d/Prog/winlibs64ucrt\_stage/custombuilt/include/libdl-win32 -march=nocona -msahf -mtune=generic -O2' CXXFLAGS='-Wno-int-conversion -march=nocona -msahf -mtune=generic -O2' LDFLAGS='-pthread -Wl,--no-insert-timestamp -Wl,--dynamicbase -Wl,--high-entropy-va -Wl,--nxcompat -Wl,--tsaware' LD=/d/Prog/winlibs64ucrt\_stage/custombuilt/share/binutils/bin/ld.exe

# Build GCC

make -j$(nproc)

# Install GCC

make install

Please replace the paths and options in the configure command with the ones that match your system configuration and requirements.

Please note that building GCC from source can take a significant amount of time and resources. Make sure your system meets the necessary requirements before proceeding.

Let me know if you need further assistance!

Source(s)

1. [github.com](https://github.com/cdecl/cdecl.github.io/tree/92f44c025fc83b494c9b98c860322f0f76f30697/_posts%2F2021-08-27-gcc-install-mac.md)

# User

I am giving you extract of: "Dump the specs file: gcc -dumpspecs > $(MINGW\_ROOT)/lib/gcc/i686-w64-mingw32/$(GCC\_VERSION)/specs" - cpp part. Can you guide me?

# Bing

Sure, I'd be happy to help. However, I'll need the extract of the cpp part from your specs file to guide you further. Could you please provide that? Once I have that information, I can help you modify it to link against UCRT instead of MSVCRT. Let me know if you have any questions!

# User

\*cpp:
%{posix:-D\_POSIX\_SOURCE} %{mthreads:-D\_MT} %{municode:-DUNICODE} %{!no-pthread:-D\_REENTRANT} %{pthread:-U\_REENTRANT}

*cpp\_options:
%(cpp\_unique\_options) %1 %{m*} %{std\*&ansi&trigraphs} %{W\*&pedantic\*} %{w} %{f\*} %{g\*:%{%:debug-level-gt(0):%{g\*} %{!fno-working-directory:-fworking-directory}}} %{O\*} %{undef} %{save-temps\*:-fpch-preprocess}

*cpp\_debug\_options:
%<dumpdir %<dumpbase %<dumpbase-ext %{d*} %:dumps()

*cpp\_unique\_options:
%{!Q:-quiet} %{nostdinc*} %{C} %{CC} %{v} %@{I\*&F\*} %{P} %I %{MD:-MD %{!o:%b.d}%{o\*:%.d%*}} %{MMD:-MMD %{!o:%b.d}%{o*:%.d%*}} %{M} %{MM} %{MF*} %{MG} %{MP} %{MQ\*} %{MT\*} %{Mmodules} %{Mno-modules} %{!E:%{!M:%{!MM:%{!MT:%{!MQ:%{MD|MMD:%{o\*:-MQ %*}}}}}}} %{remap} %{%:debug-level-gt(2):-dD} %{!iplugindir*:%{fplugin\*:%:find-plugindir()}} %{H} %C %{D\*&U\*&A\*} %{i\*} %Z %i %{E|M|MM:%W{o\*}}

# Bing

Sure, I can guide you on how to modify the cpp part of your specs file to link against UCRT instead of MSVCRT. Here's how you can do it:

1. **Add `-D\_UCRT` to the `cpp` section**:

\*cpp:

%{posix:-D\_POSIX\_SOURCE} %{mthreads:-D\_MT} %{municode:-DUNICODE} %{!no-pthread:-D\_REENTRANT} %{pthread:-U\_REENTRANT} -D\_UCRT

This will prevent undefined references linking errors for the scanf functions family.

1. **Replace `-lmsvcrt` with `-lucrt` in the `libgcc` section**:

You'll need to find the libgcc section in your specs file and replace -lmsvcrt with -lucrt. This will change the way MinGW GCC operates to link against UCRT instead of MSVCRT.

Please note that the libgcc section is not included in the extract you provided. You'll need to find this section in your full specs file.

Let me know if you need further assistance!

# User

My cc1plus extract from specs is as under:

\*cc1plus:

\*link\_gcc\_c\_sequence:
%G %{!nolibc:%L %G}

As you can see there is nothing underneath it. Then another section starts: \*link\_gcc\_c\_sequence:

# Bing

Thank you for providing the cc1plus and link\_gcc\_c\_sequence sections from your specs file. However, these sections are not the ones we need to modify for linking against UCRT instead of MSVCRT.

The section we need to modify is the \*libgcc section. This section is where the standard libraries are specified for linking. You'll need to find this section in your full specs file and replace -lmsvcrt with -lucrt.

If you can't find the \*libgcc section in your specs file, it might be because your GCC installation is using a different specs file or it's using default specs embedded in the GCC executables. In that case, you might need to create a new specs file with the necessary modifications.

Please let me know if you need further assistance! I'm here to help.

# User

Here is the extract of \*libgcc:

\*libgcc:
%{mthreads:-lmingwthrd} -lmingw32 %{static|static-libgcc:-lgcc -lgcc\_eh} %{!static: %{!static-libgcc: %{!shared: %{!shared-libgcc:-lgcc -lgcc\_eh} %{shared-libgcc:-lgcc\_s -lgcc} } %{shared:-lgcc\_s -lgcc} } } -lmoldname -lmingwex -lmsvcrt -lkernel32

Where do I change?

# Bing

In the \*libgcc section of your specs file, you need to replace -lmsvcrt with -lucrt. Here's how the modified \*libgcc section should look:

\*libgcc:

%{mthreads:-lmingwthrd} -lmingw32 %{static|static-libgcc:-lgcc -lgcc\_eh} %{!static: %{!static-libgcc: %{!shared: %{!shared-libgcc:-lgcc -lgcc\_eh} %{shared-libgcc:-lgcc\_s -lgcc} } %{shared:-lgcc\_s -lgcc} } } -lmoldname -lmingwex -lucrt -lkernel32

This change will make MinGW GCC link against UCRT instead of MSVCRT. Let me know if you need further assistance!

Source(s)

1. [github.com](https://github.com/webos21/webos21.github.com/tree/b790145827b5f2fe7fb9a26a72fb2dd1f9f785ac/_posts%2F2019-07-01-mingw-w64.md)

# User

Okay lets go one by one. First I change cpp as stated by you and then \*libgcc. What next?

# Bing

Great! After modifying the cpp and \*libgcc sections in your specs file, the next steps are to clone the GCC repository and build GCC. Here's how you can do it:

1. **Clone the GCC repository**:

# Clone the GCC repository

git clone https://github.com/gcc-mirror/gcc.git

1. **Navigate to the GCC directory**:

# Navigate to the GCC directory

cd gcc

1. **Create a build directory and navigate to it**:

# Create a build directory and navigate to it

mkdir build

cd build

1. **Configure the build**:

# Configure the build

../configure --prefix=/R/winlibs64ucrt\_stage/inst\_gcc-13.2.0/share/gcc --build=x86\_64-w64-mingw32 --host=x86\_64-w64-mingw32 --enable-offload-targets=nvptx-none --with-pkgversion='MinGW-W64 x86\_64-ucrt-posix-seh, built by Brecht Sanders, r4' --with-tune=generic --enable-checking=release --enable-threads=posix --disable-sjlj-exceptions --disable-libunwind-exceptions --disable-serial-configure --disable-bootstrap --enable-host-shared --enable-plugin --disable-default-ssp --disable-rpath --disable-libstdcxx-debug --disable-version-specific-runtime-libs --with-stabs --disable-symvers --enable-languages=c,c++,fortran,lto,objc,obj-c++ --disable-gold --disable-nls --disable-stage1-checking --disable-win32-registry --disable-multilib --enable-ld --enable-libquadmath --enable-libada --enable-libssp --enable-libstdcxx --enable-lto --enable-fully-dynamic-string --enable-libgomp --enable-graphite --enable-mingw-wildcard --enable-libstdcxx-time --enable-libstdcxx-pch --with-mpc=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-mpfr=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-gmp=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-isl=/d/Prog/winlibs64ucrt\_stage/custombuilt --disable-libstdcxx-backtrace --enable-install-libiberty --enable-\_\_cxa\_atexit --without-included-gettext --with-diagnostics-color=auto --enable-clocale=generic --with-libiconv --with-system-zlib --with-build-sysroot=/R/winlibs64ucrt\_stage/gcc-13.2.0/build\_mingw/mingw-w64 CFLAGS='-I/d/Prog/winlibs64ucrt\_stage/custombuilt/include/libdl-win32 -march=nocona -msahf -mtune=generic -O2' CXXFLAGS='-Wno-int-conversion -march=nocona -msahf -mtune=generic -O2' LDFLAGS='-pthread -Wl,--no-insert-timestamp -Wl,--dynamicbase -Wl,--high-entropy-va -Wl,--nxcompat -Wl,--tsaware' LD=/d/Prog/winlibs64ucrt\_stage/custombuilt/share/binutils/bin/ld.exe

Please replace the paths and options in the configure command with the ones that match your system configuration and requirements.

1. **Build GCC**:

# Build GCC

make -j$(nproc)

1. **Install GCC**:

# Install GCC

make install

Please note that building GCC from source can take a significant amount of time and resources. Make sure your system meets the necessary requirements before proceeding.

Let me know if you need further assistance!

Source(s)

1. [github.com](https://github.com/cdecl/cdecl.github.io/tree/92f44c025fc83b494c9b98c860322f0f76f30697/_posts%2F2021-08-27-gcc-install-mac.md)

# User

But how does specs file changes will be taken care of with above command? Also I am building gcc 14 where as configuration mentions 13.2. I want to build gcc 14 under msys shell in /opt/gcc directory. Can you modify the above configuration parameters to do that?

# Bing

The changes you made to the specs file will be used by GCC when it's compiling and linking your programs. The specs file is a configuration file that GCC reads each time it's run. It tells GCC what libraries to link against, among other things. By modifying the specs file, you're telling GCC to link against UCRT instead of MSVCRT.

As for building GCC 14 under the msys shell in the /opt/gcc directory, you can modify the configure command as follows:

# Configure the build

../configure --prefix=/opt/gcc --build=x86\_64-w64-mingw32 --host=x86\_64-w64-mingw32 --enable-offload-targets=nvptx-none --with-pkgversion='MinGW-W64 x86\_64-ucrt-posix-seh, built by Brecht Sanders, r4' --with-tune=generic --enable-checking=release --enable-threads=posix --disable-sjlj-exceptions --disable-libunwind-exceptions --disable-serial-configure --disable-bootstrap --enable-host-shared --enable-plugin --disable-default-ssp --disable-rpath --disable-libstdcxx-debug --disable-version-specific-runtime-libs --with-stabs --disable-symvers --enable-languages=c,c++,fortran,lto,objc,obj-c++ --disable-gold --disable-nls --disable-stage1-checking --disable-win32-registry --disable-multilib --enable-ld --enable-libquadmath --enable-libada --enable-libssp --enable-libstdcxx --enable-lto --enable-fully-dynamic-string --enable-libgomp --enable-graphite --enable-mingw-wildcard --enable-libstdcxx-time --enable-libstdcxx-pch --with-mpc=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-mpfr=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-gmp=/d/Prog/winlibs64ucrt\_stage/custombuilt --with-isl=/d/Prog/winlibs64ucrt\_stage/custombuilt --disable-libstdcxx-backtrace --enable-install-libiberty --enable-\_\_cxa\_atexit --without-included-gettext --with-diagnostics-color=auto --enable-clocale=generic --with-libiconv --with-system-zlib --with-build-sysroot=/R/winlibs64ucrt\_stage/gcc-13.2.0/build\_mingw/mingw-w64 CFLAGS='-I/d/Prog/winlibs64ucrt\_stage/custombuilt/include/libdl-win32 -march=nocona -msahf -mtune=generic -O2' CXXFLAGS='-Wno-int-conversion -march=nocona -msahf -mtune=generic -O2' LDFLAGS='-pthread -Wl,--no-insert-timestamp -Wl,--dynamicbase -Wl,--high-entropy-va -Wl,--nxcompat -Wl,--tsaware' LD=/d/Prog/winlibs64ucrt\_stage/custombuilt/share/binutils/bin/ld.exe

This command will configure the build to install GCC in the /opt/gcc directory. Please replace the paths and options in the configure command with the ones that match your system configuration and requirements.

Let me know if you need further assistance!

Source(s)

1. [github.com](https://github.com/cdecl/cdecl.github.io/tree/92f44c025fc83b494c9b98c860322f0f76f30697/_posts%2F2021-08-27-gcc-install-mac.md)

# User

I see references to: /d/Prog/winlibs64ucrt\_stage at several places in configuration command. Will that not change if I build gcc at /opt/gcc location?

# Bing

The /d/Prog/winlibs64ucrt\_stage path in the configuration command refers to the location where the dependencies for building GCC are installed, not the location where GCC itself will be installed. These dependencies include libraries such as MPC, MPFR, GMP, and ISL.

The --prefix=/opt/gcc option in the configuration command is what determines where GCC will be installed. So, even if the dependencies are located in /d/Prog/winlibs64ucrt\_stage, GCC will still be installed in /opt/gcc as specified.

However, if the dependencies are located in a different directory on your system, you should replace /d/Prog/winlibs64ucrt\_stage with the correct path. For example, if your dependencies are located in /path/to/dependencies, the configuration command would look like this:

../configure --prefix=/opt/gcc --build=x86\_64-w64-mingw32 --host=x86\_64-w64-mingw32 --enable-offload-targets=nvptx-none --with-pkgversion='MinGW-W64 x86\_64-ucrt-posix-seh, built by Brecht Sanders, r4' --with-tune=generic --enable-checking=release --enable-threads=posix --disable-sjlj-exceptions --disable-libunwind-exceptions --disable-serial-configure --disable-bootstrap --enable-host-shared --enable-plugin --disable-default-ssp --disable-rpath --disable-libstdcxx-debug --disable-version-specific-runtime-libs --with-stabs --disable-symvers --enable-languages=c,c++,fortran,lto,objc,obj-c++ --disable-gold --disable-nls --disable-stage1-checking --disable-win32-registry --disable-multilib --enable-ld --enable-libquadmath --enable-libada --enable-libssp --enable-libstdcxx --enable-lto --enable-fully-dynamic-string --enable-libgomp --enable-graphite --enable-mingw-wildcard --enable-libstdcxx-time --enable-libstdcxx-pch --with-mpc=/path/to/dependencies --with-mpfr=/path/to/dependencies --with-gmp=/path/to/dependencies --with-isl=/path/to/dependencies --disable-libstdcxx-backtrace --enable-install-libiberty --enable-\_\_cxa\_atexit --without-included-gettext --with-diagnostics-color=auto --enable-clocale=generic --with-libiconv --with-system-zlib --with-build-sysroot=/R/winlibs64ucrt\_stage/gcc-13.2.0/build\_mingw/mingw-w64 CFLAGS='-I/path/to/dependencies/include/libdl-win32 -march=nocona -msahf -mtune=generic -O2' CXXFLAGS='-Wno-int-conversion -march=nocona -msahf -mtune=generic -O2' LDFLAGS='-pthread -Wl,--no-insert-timestamp -Wl,--dynamicbase -Wl,--high-entropy-va -Wl,--nxcompat -Wl,--tsaware' LD=/path/to/dependencies/share/binutils/bin/ld.exe

Let me know if you need further assistance! I'm here to help.

Source(s)

1. [github.com](https://github.com/cdecl/cdecl.github.io/tree/92f44c025fc83b494c9b98c860322f0f76f30697/_posts%2F2021-08-27-gcc-install-mac.md)

# User

What are all the dependencies you can see in configuration command? Also, whether all these are available through github only or where to install them?

# Bing

The dependencies mentioned in the configuration command are libraries that GCC requires for building. Here are the dependencies:

1. **MPC (Multiple Precision Complex library)**: This is a complex arithmetic library with arbitrarily high precision.
2. **MPFR (Multiple Precision Floating-Point Reliable library)**: This is a library for multiple-precision floating-point computations with correct rounding.
3. **GMP (GNU Multiple Precision Arithmetic Library)**: This is a free library for arbitrary precision arithmetic, operating on signed integers, rational numbers, and floating-point numbers.
4. **ISL (Integer Set Library)**: This is a library for manipulating sets and relations of integer points bounded by linear constraints.

These libraries are not necessarily available on GitHub. They are open-source libraries and can be downloaded from their respective official websites or other trusted sources. Here are the official websites for these libraries:

* MPC: <http://www.multiprecision.org/mpc/>
* MPFR: <https://www.mpfr.org/>
* GMP: <https://gmplib.org/>
* ISL: <http://isl.gforge.inria.fr/>

You can download the source code from these websites and install them following the instructions provided in their documentation. Please note that you might need certain permissions to install these libraries depending on your system configuration.

Let me know if you need further assistance!

# User

I started - As MPC requires MPFR and GMP, I downloaded them first. When building MPFR during ./configure in Msys, I encountered error gmp not found. So I started building gmp first. I issued command ./configure --prefix=/opt/gmp in msys bash shell prompt. It ended with error:

checking for suitable m4... configure: error: No usable m4 in $PATH or /usr/5bin (see config.log for reasons).

Whether based on above, can you tell me what wrong am I doing?

# Bing

The error message you're seeing indicates that the m4 macro processor is not found in your system's PATH. m4 is a tool that's required for building many open-source projects, including GMP.

Here are a few possible solutions:

1. **Install m4 using the package manager of MSYS2**: If you're using MSYS2, you can install m4 using the package manager pacman with the command pacman -S m4[**1**](https://stackoverflow.com/questions/11368989/no-usable-m4-in-path-or-usr5bin).
2. **Install m4 using the Cygwin installer**: If you're using Cygwin, the setup installer has a working package of m4. You can select it from the list and install[**1**](https://stackoverflow.com/questions/11368989/no-usable-m4-in-path-or-usr5bin).
3. **Check your PATH variable**: Make sure the directory containing m4 is in your PATH. You can check your PATH by typing echo $PATH in your shell. If the directory containing m4 is not in your PATH, you can add it with the command export PATH=/path/to/m4:$PATH, replacing /path/to/m4 with the actual directory path[**1**](https://stackoverflow.com/questions/11368989/no-usable-m4-in-path-or-usr5bin).
4. **Reinstall m4**: If m4 is installed but not working correctly, you might need to reinstall it. You can usually do this through your system's package manager[**1**](https://stackoverflow.com/questions/11368989/no-usable-m4-in-path-or-usr5bin).

Please try these solutions and let me know if you need further assistance!

Source(s)

1. [windows - No usable M4 in $PATH or /usr5bin - Stack Overflow](https://stackoverflow.com/questions/11368989/no-usable-m4-in-path-or-usr5bin)

2. [Cygwin crashing when building gmp library - Stack Overflow](https://stackoverflow.com/questions/11488158/cygwin-crashing-when-building-gmp-library)

3. [linux下安装gmp遇到 configure:error:no usable m4 in$path or /user/5bin解决方案 ...](https://www.cnblogs.com/yg-Mark/p/12911879.html)

4. [c++ - Error when configuring gmp - Stack Overflow](https://stackoverflow.com/questions/7902894/error-when-configuring-gmp)

# User

Okay, I did that (installed m4 using pacman in msys2) - ./configure ended successfully with following command:

Version: GNU MP 6.3.0
Host type: x86\_64-pc-msys
ABI: 64
Install prefix: /opt/gmp
Compiler: gcc
Static libraries: yes
Shared libraries: no

How can I also build with Shared Libraries as yes?

# Bing

To build GMP with shared libraries, you can add the --enable-shared option and remove the --disable-static option when running the configure command. Here's how you can do it:

./configure --prefix=/opt/gmp --enable-shared

This command will configure the build to install GMP in the /opt/gmp directory and enable the building of shared libraries[**1**](https://gmplib.org/manual/Build-Options).

After running the configure command with the --enable-shared option, you can proceed with the make and make install commands to build and install GMP.

Please note that building shared libraries might require additional dependencies. If you encounter any issues, please let me know! I'm here to help.

Source(s)

1. [Build Options (GNU MP 6.3.0) - gmplib.org](https://gmplib.org/manual/Build-Options)

2. [How to install GCC piece by piece with GMP, MPFR, MPC, ELF, without ...](https://stackoverflow.com/questions/9450394/how-to-install-gcc-piece-by-piece-with-gmp-mpfr-mpc-elf-without-shared-libra)

3. [Notes for Particular Systems (GNU MP 6.3.0) - gmplib.org](https://gmplib.org/manual/Notes-for-Particular-Systems)

4. [Build GMP on 64bit Windows - Stack Overflow](https://stackoverflow.com/questions/4711315/build-gmp-on-64bit-windows)

5. <http://gcc.gnu.org/wiki/InstallingGCC>

6. <ftp://gcc.gnu.org/pub/gcc/infrastructure/>

7. <ftp://gcc.gnu.org/pub/gcc/infrastructure/gmp-4.3.2.tar.bz2>

# User

I issued:

./configure prefix=/opt/gmp --enable-shared --disable-static

because I cannot enable both shared and static together. Is it possible. I had to do that because otherwise ./configure cannot proceed. However, even after this command, I still see:

Version: GNU MP 6.3.0
Host type: x86\_64-pc-msys
ABI: 64
Install prefix: /opt/gmp
Compiler: gcc
Static libraries: yes
Shared libraries: no

As you can see Shared libraries: No. Why?

# Bing

The configure script should be able to build both static and shared libraries by default unless explicitly disabled[**2**](https://gmplib.org/manual/Notes-for-Particular-Systems). However, there could be a few reasons why shared libraries are not being built:

1. **Missing Dependencies**: Building shared libraries may require additional dependencies that are not currently installed on your system[**2**](https://gmplib.org/manual/Notes-for-Particular-Systems).
2. **Configuration Issues**: There could be issues with the configuration options you're using. The --prefix option should have two hyphens (--). So, the correct command would be ./configure --prefix=/opt/gmp --enable-shared --disable-static.
3. **System-Specific Issues**: Some systems have known issues with building shared libraries. For example, on certain AIX systems, shared libraries are disabled by default[**4**](https://gmplib.org/manual/Known-Build-Problems).

If you're still having trouble, you could check the config.log file that's generated by the configure script. This file contains detailed information about the checks that configure performed, and it might include clues about why shared libraries are not being built.

Let me know if you need further assistance!

Source(s)

1. [Build Options (GNU MP 6.3.0) - gmplib.org](https://gmplib.org/manual/Build-Options)

2. [Notes for Particular Systems (GNU MP 6.3.0) - gmplib.org](https://gmplib.org/manual/Notes-for-Particular-Systems)

3. [c - gmp shared libraries not found - Stack Overflow](https://stackoverflow.com/questions/7225990/gmp-shared-libraries-not-found)

4. [Known Build Problems (GNU MP 6.3.0) - gmplib.org](https://gmplib.org/manual/Known-Build-Problems)

# User

I could build gmp. Now on to mpfr (because it is required for MPC along with GMP). How I can point to /opt/gmp while building mpfr because gmp is a dependency for mpfr (To remind you, I am on msys2 shell)

# Bing

Great, you've successfully built GMP! Now, to build MPFR with GMP as a dependency, you can specify the location of your GMP installation using the --with-gmp option during the configuration step[**1**](https://stackoverflow.com/questions/62135116/msys2-mingw64-to-build-gmp-mpfr-on-windows-as-static-library-and-link-them-in-m). Here's how you can do it:

1. **Download and extract the MPFR source code**:

wget http://www.mpfr.org/mpfr-current/mpfr-4.1.0.tar.gz

tar xzf mpfr-4.1.0.tar.gz

cd mpfr-4.1.0

1. **Configure the build to use the GMP library you've installed in `/opt/gmp`**:

./configure --prefix=/opt/mpfr --with-gmp=/opt/gmp

1. **Build and install MPFR**:

make

make check

make install

This will build MPFR with GMP as a dependency and install it in the /opt/mpfr directory.

Let me know if you need further assistance!

Source(s)

1. [MSYS2 MinGW64 to build GMP/MPFR on Windows as static library, and link ...](https://stackoverflow.com/questions/62135116/msys2-mingw64-to-build-gmp-mpfr-on-windows-as-static-library-and-link-them-in-m)

2. [c - How to build GMP in MSYS2? - Stack Overflow](https://stackoverflow.com/questions/63217181/how-to-build-gmp-in-msys2)

3. [GitHub - emphasis87/libmpfr-msys2-mingw64: MPFR library built for ...](https://github.com/emphasis87/libmpfr-msys2-mingw64)

4. [GNU MPFR Third-Party Packages and Ports](https://www.mpfr.org/ports.html)

# User

During make check for mpfr, I found following error:

FAIL: tsprintf.exe

# ============================================================================ Testsuite summary for MPFR 4.2.1

# TOTAL: 198

# PASS: 197

# SKIP: 0

# XFAIL: 0

# FAIL: 1

# XPASS: 0

# ERROR: 0

# ============================================================================ See tests/test-suite.log

make[3]: \*\*\* [Makefile:3026: test-suite.log] Error 1
make[3]: Leaving directory '/home/juhi123/Downloads/mpfr-4.2.1/tests'
make[2]: \*\*\* [Makefile:3134: check-TESTS] Error 2
make[2]: Leaving directory '/home/juhi123/Downloads/mpfr-4.2.1/tests'
make[1]: \*\*\* [Makefile:4586: check-am] Error 2
make[1]: Leaving directory '/home/juhi123/Downloads/mpfr-4.2.1/tests'
make: \*\*\* [Makefile:524: check-recursive] Error 1

# Bing

The error you're seeing is from the tsprintf.exe test failing during the make check step of the MPFR build process. This test failure is known to occur on some systems due to compatibility issues with the MSVC runtime[**3**](https://www.mpfr.org/faq.html).

While it's generally a good idea to investigate and resolve test failures, in this case, if you're only seeing a failure with tsprintf.exe and all other tests are passing, it's likely that this failure won't impact the functionality of MPFR for building GCC[**3**](https://www.mpfr.org/faq.html).

You can proceed with the make install step to install MPFR. However, please be aware that proceeding despite the test failure means you're accepting a small risk that there could be an issue with your MPFR installation. If you encounter issues later on when using GCC, you might need to revisit this.

If you want to investigate this further, you could check the tests/test-suite.log file, which should contain more detailed information about the test failure[**3**](https://www.mpfr.org/faq.html).

Let me know if you need further assistance! I'm here to help.

Source(s)

1. [mpfr-2.4.1/INSTALL - native\_client/nacl-gcc - Git at Google](https://chromium.googlesource.com/native_client/nacl-gcc/+/master/mpfr-2.4.1/INSTALL)

2. [Cannot configure gcc - mpfr not found - Stack Overflow](https://stackoverflow.com/questions/9297933/cannot-configure-gcc-mpfr-not-found)

3. [Frequently Asked Questions about GNU MPFR](https://www.mpfr.org/faq.html)

# User

I checked log. It is as under:

repl-vsnprintf.c:388: GNU MP assertion failed: len < total\_width

FAIL tsprintf.exe (exit status: 127)

# Bing

The error message you're seeing is from the tsprintf.exe test failing during the make check step of the MPFR build process. This test failure is known to occur on some systems due to compatibility issues[**3**](https://stackoverflow.com/questions/9450394/how-to-install-gcc-piece-by-piece-with-gmp-mpfr-mpc-elf-without-shared-libra).

While it's generally a good idea to investigate and resolve test failures, in this case, if you're only seeing a failure with tsprintf.exe and all other tests are passing, it's likely that this failure won't impact the functionality of MPFR for building GCC[**3**](https://stackoverflow.com/questions/9450394/how-to-install-gcc-piece-by-piece-with-gmp-mpfr-mpc-elf-without-shared-libra).

You can proceed with the make install step to install MPFR. However, please be aware that proceeding despite the test failure means you're accepting a small risk that there could be an issue with your MPFR installation. If you encounter issues later on when using GCC, you might need to revisit this.

If you want to investigate this further, you could check the tests/test-suite.log file, which should contain more detailed information about the test failure[**3**](https://stackoverflow.com/questions/9450394/how-to-install-gcc-piece-by-piece-with-gmp-mpfr-mpc-elf-without-shared-libra).

Let me know if you need further assistance! I'm here to help.

Source(s)

1. [[SOLVED] C++ runtime error: Assertion Failed - Ubuntu Forums](https://ubuntuforums.org/showthread.php?t=2160022)

2. [c - MPFR Program Crashes With High Precision - Stack Overflow](https://stackoverflow.com/questions/32255374/mpfr-program-crashes-with-high-precision)

3. [How to install GCC piece by piece with GMP, MPFR, MPC, ELF, without ...](https://stackoverflow.com/questions/9450394/how-to-install-gcc-piece-by-piece-with-gmp-mpfr-mpc-elf-without-shared-libra)

4. [JOSS: MPFR assertion failed · Issue #36 - GitHub](https://github.com/AngryMaciek/hypercomplex/issues/36)

5. <http://gcc.gnu.org/wiki/InstallingGCC>

6. <ftp://gcc.gnu.org/pub/gcc/infrastructure/>

7. <http://www.holoborodko.com/pavel/mpfr>

# User

Can you search specifically for this error in building mpfr:

repl-vsnprintf.c:388: GNU MP assertion failed: len < total\_width

# Bing

The error repl-vsnprintf.c:388: GNU MP assertion failed: len < total\_width you're encountering during the make check step of the MPFR build process seems to be related to the vsnprintf function[**2**](https://github.com/ShiftMediaProject/gmp/blob/master/printf/repl-vsnprintf.c). This function is part of the GNU Multiple Precision Arithmetic Library (GMP), which MPFR depends on[**2**](https://github.com/ShiftMediaProject/gmp/blob/master/printf/repl-vsnprintf.c).

This error might occur if the vsnprintf function is called with arguments that do not meet its requirements[**2**](https://github.com/ShiftMediaProject/gmp/blob/master/printf/repl-vsnprintf.c). For example, if the size of the output string (len) is not less than the total width of the formatted output string (total\_width), this assertion would fail[**2**](https://github.com/ShiftMediaProject/gmp/blob/master/printf/repl-vsnprintf.c).

One possible solution to this issue is to ensure that the vsnprintf function is called correctly[**2**](https://github.com/ShiftMediaProject/gmp/blob/master/printf/repl-vsnprintf.c). This might involve checking the arguments passed to vsnprintf in your code or in the libraries you're using[**2**](https://github.com/ShiftMediaProject/gmp/blob/master/printf/repl-vsnprintf.c).

Another possible solution is to update or reinstall GMP[**3**](https://stackoverflow.com/questions/72437147/how-to-prevent-gcc-from-compiling-error-prone-vsnprintf). This error might be due to a bug or compatibility issue in the version of GMP you're using[**3**](https://stackoverflow.com/questions/72437147/how-to-prevent-gcc-from-compiling-error-prone-vsnprintf).

If you're still having trouble, you could check the tests/test-suite.log file, which should contain more detailed information about the test failure[**3**](https://stackoverflow.com/questions/72437147/how-to-prevent-gcc-from-compiling-error-prone-vsnprintf).

Please note that these are general suggestions based on the error message you provided. The exact solution might depend on your specific system configuration and the exact cause of the error[**3**](https://stackoverflow.com/questions/72437147/how-to-prevent-gcc-from-compiling-error-prone-vsnprintf).

Let me know if you need further assistance! I'm here to help.

Source(s)

1. [c - How do you call vsnprintf() safely? - Stack Overflow](https://stackoverflow.com/questions/37788305/how-do-you-call-vsnprintf-safely)

2. [gmp/repl-vsnprintf.c at master · ShiftMediaProject/gmp](https://github.com/ShiftMediaProject/gmp/blob/master/printf/repl-vsnprintf.c)

3. [c - How to prevent GCC from compiling error-prone vsnprintf? - Stack ...](https://stackoverflow.com/questions/72437147/how-to-prevent-gcc-from-compiling-error-prone-vsnprintf)

4. [[MPFR] assertion failed: size\_s1 >= m #22122 - GitHub](https://github.com/microsoft/vcpkg/issues/22122)

# User

Does this error will affect gcc building? Can you search?

# Bing

The error repl-vsnprintf.c:388: GNU MP assertion failed: len < total\_width you're encountering during the make check step of the MPFR build process seems to be related to the vsnprintf function[**2**](https://linux.die.net/man/3/vsnprintf). This function is part of the GNU Multiple Precision Arithmetic Library (GMP), which MPFR depends on[**2**](https://linux.die.net/man/3/vsnprintf).

This error might occur if the vsnprintf function is called with arguments that do not meet its requirements[**2**](https://linux.die.net/man/3/vsnprintf). For example, if the size of the output string (len) is not less than the total width of the formatted output string (total\_width), this assertion would fail[**2**](https://linux.die.net/man/3/vsnprintf).

One possible solution to this issue is to ensure that the vsnprintf function is called correctly[**2**](https://linux.die.net/man/3/vsnprintf). This might involve checking the arguments passed to vsnprintf in your code or in the libraries you're using[**2**](https://linux.die.net/man/3/vsnprintf).

Another possible solution is to update or reinstall GMP[**3**](https://stackoverflow.com/questions/17351874/is-vsnprintf-s-an-appropriate-replacement-for-deprecated-vsnprintf). This error might be due to a bug or compatibility issue in the version of GMP you're using[**3**](https://stackoverflow.com/questions/17351874/is-vsnprintf-s-an-appropriate-replacement-for-deprecated-vsnprintf).

If you're still having trouble, you could check the tests/test-suite.log file, which should contain more detailed information about the test failure[**3**](https://stackoverflow.com/questions/17351874/is-vsnprintf-s-an-appropriate-replacement-for-deprecated-vsnprintf).

Please note that these are general suggestions based on the error message you provided. The exact solution might depend on your specific system configuration and the exact cause of the error[**3**](https://stackoverflow.com/questions/17351874/is-vsnprintf-s-an-appropriate-replacement-for-deprecated-vsnprintf).

Let me know if you need further assistance! I'm here to help.

Source(s)

1. [c - How do you call vsnprintf() safely? - Stack Overflow](https://stackoverflow.com/questions/37788305/how-do-you-call-vsnprintf-safely)

2. [vsnprintf(3): formatted output conversion - Linux man page](https://linux.die.net/man/3/vsnprintf)

3. [c - Is vsnprintf\_s an appropriate replacement for deprecated vsnprintf ...](https://stackoverflow.com/questions/17351874/is-vsnprintf-s-an-appropriate-replacement-for-deprecated-vsnprintf)

4. [vprintf, vfprintf, vsprintf, vsnprintf, vprintf\_s, vfprintf\_s, vsprintf ...](https://en.cppreference.com/w/c/io/vfprintf)

# User

Okay Can you give me test C code to test this vsprintf using gmp built at /opt/gmp?

# Bing

Sure, here is a simple C program that uses vsnprintf from the GMP library. This program prints a floating-point number with a specified precision.

#include <stdio.h>

#include <gmp.h>

int main() {

mpf\_t pi;

mpf\_init\_set\_str(pi, "3.14159265358979323846", 10);

char buffer[100];

mp\_exp\_t exp;

char \*str = mpf\_get\_str(NULL, &exp, 10, 0, pi);

int precision = 10; // Change this to the desired precision

snprintf(buffer, sizeof(buffer), "%%.%dF", precision);

printf(buffer, mpf\_get\_d(pi));

mpf\_clear(pi);

return 0;

}

To compile and run this program with the GMP library installed in /opt/gmp, you can use the following commands:

gcc -I/opt/gmp/include -L/opt/gmp/lib -lgmp test.c -o test

./test

Please replace test.c with the name of your C file. This program should print the value of pi to the console with the specified precision.

Let me know if you need further assistance! I'm here to help.

# User

But I don't see any vsprintf above.

# Bing

I apologize for the confusion. You're correct, the previous example used snprintf, not vsnprintf. Here's a simple example of how to use vsnprintf:

#include <stdio.h>

#include <stdarg.h>

void print\_formatted(const char \*format, ...) {

char buffer[100];

va\_list args;

va\_start(args, format);

vsnprintf(buffer, sizeof(buffer), format, args);

va\_end(args);

printf("%s\n", buffer);

}

int main() {

print\_formatted("Hello, %s!", "World");

return 0;

}

In this example, the print\_formatted function uses vsnprintf to format a string with a variable number of arguments. The va\_start and va\_end functions are used to handle the variable argument list.

You can compile and run this program with the following commands:

gcc test.c -o test

./test

Please replace test.c with the name of your C file. This program should print "Hello, World!" to the console.

Let me know if you need further assistance! I'm here to help.

# User

Is vsnprintf same as vsprintf?

# Bing

No, vsnprintf and vsprintf are not the same. They are both functions in the C standard library that format a variable argument list and write the output to a string, but they behave differently[**3**](https://gamedev.net/forums/topic/613821-sprintf-vs-vsprintf/4877697/)[**4**](https://stackoverflow.com/questions/46485639/what-is-the-difference-between-vsnprintf-and-vsprintf-s):

* vsprintf: This function writes the output to a character string buffer[**5**](https://en.cppreference.com/w/c/io/vfprintf). It does not check the size of the buffer, which can lead to buffer overflow if the formatted output string is larger than the buffer[**5**](https://en.cppreference.com/w/c/io/vfprintf).
* vsnprintf: This function also writes the output to a character string buffer, but it takes an additional argument specifying the size of the buffer[**3**](https://gamedev.net/forums/topic/613821-sprintf-vs-vsprintf/4877697/). It writes at most size - 1 characters to the buffer, ensuring that the output is null-terminated and preventing buffer overflow[**3**](https://gamedev.net/forums/topic/613821-sprintf-vs-vsprintf/4877697/).

In general, vsnprintf is safer to use than vsprintf because it prevents buffer overflow, which can lead to security vulnerabilities[**1**](https://linux.die.net/man/3/vsnprintf)[**2**](https://www.oreilly.com/library/view/c-in-a/0596006977/re255.html).

Source(s)

1. [vsnprintf(3): formatted output conversion - Linux man page](https://linux.die.net/man/3/vsnprintf)

2. [vfprintf, vprintf, vsnprintf, vsprintf - C in a Nutshell [Book]](https://www.oreilly.com/library/view/c-in-a/0596006977/re255.html)

3. [sprintf vs vsprintf - For Beginners - GameDev.net](https://gamedev.net/forums/topic/613821-sprintf-vs-vsprintf/4877697/)

4. [What is the difference between vsnprintf and vsprintf\_s?](https://stackoverflow.com/questions/46485639/what-is-the-difference-between-vsnprintf-and-vsprintf-s)

5. [vprintf, vfprintf, vsprintf, vsnprintf, vprintf\_s, vfprintf\_s, vsprintf ...](https://en.cppreference.com/w/c/io/vfprintf)

# User

But I need test code for vsprintf not vsnprintf.

# Bing

I apologize for the confusion. Here's a simple example of how to use vsprintf:

#include <stdio.h>

#include <stdarg.h>

void print\_formatted(const char \*format, ...) {

char buffer[100];

va\_list args;

va\_start(args, format);

vsprintf(buffer, format, args);

va\_end(args);

printf("%s\n", buffer);

}

int main() {

print\_formatted("Hello, %s!", "World");

return 0;

}

In this example, the print\_formatted function uses vsprintf to format a string with a variable number of arguments. The va\_start and va\_end functions are used to handle the variable argument list.

You can compile and run this program with the following commands:

gcc test.c -o test

./test

Please replace test.c with the name of your C file. This program should print "Hello, World!" to the console.

Please note that vsprintf does not check the size of the buffer and can cause a buffer overflow if the formatted output string is larger than the buffer. It's generally safer to use vsnprintf, which takes an additional argument specifying the size of the buffer.

Let me know if you need further assistance! I'm here to help.

# User

See this test program for vsprintf ran without errors. So I am confused, if vsprintf test is successful, why during building mpfr, I got the Failed: vsprinf error during make check command (Just see my messages above).

# Bing

The vsprintf test in your standalone program and the vsprintf test in the MPFR build process are not exactly the same. The standalone test is a simple use case, while the test during the MPFR build process is likely more complex and could be involving other parts of the MPFR library[**1**](https://github.com/microsoft/vcpkg/issues/19353).

The error repl-vsnprintf.c:388: GNU MP assertion failed: len < total\_width you're encountering during the make check step of the MPFR build process seems to be related to the vsnprintf function[**2**](https://stackoverflow.com/questions/32740172/unresolved-stdio-common-vsprintf-s-what-library-has-this). This function is part of the GNU Multiple Precision Arithmetic Library (GMP), which MPFR depends on[**2**](https://stackoverflow.com/questions/32740172/unresolved-stdio-common-vsprintf-s-what-library-has-this).

This error might occur if the vsnprintf function is called with arguments that do not meet its requirements[**2**](https://stackoverflow.com/questions/32740172/unresolved-stdio-common-vsprintf-s-what-library-has-this). For example, if the size of the output string (len) is not less than the total width of the formatted output string (total\_width), this assertion would fail[**2**](https://stackoverflow.com/questions/32740172/unresolved-stdio-common-vsprintf-s-what-library-has-this).

One possible solution to this issue is to ensure that the vsnprintf function is called correctly[**2**](https://stackoverflow.com/questions/32740172/unresolved-stdio-common-vsprintf-s-what-library-has-this). This might involve checking the arguments passed to vsnprintf in your code or in the libraries you're using[**2**](https://stackoverflow.com/questions/32740172/unresolved-stdio-common-vsprintf-s-what-library-has-this).

Another possible solution is to update or reinstall GMP[**3**](https://github.com/microsoft/vcpkg/issues/23641). This error might be due to a bug or compatibility issue in the version of GMP you're using[**3**](https://github.com/microsoft/vcpkg/issues/23641).

If you're still having trouble, you could check the tests/test-suite.log file, which should contain more detailed information about the test failure[**3**](https://github.com/microsoft/vcpkg/issues/23641).

Please note that these are general suggestions based on the error message you provided. The exact solution might depend on your specific system configuration and the exact cause of the error[**3**](https://github.com/microsoft/vcpkg/issues/23641).

Let me know if you need further assistance! I'm here to help.

Source(s)

1. [[mpfr] x64-windows build failure · Issue #19353 · microsoft/vcpkg](https://github.com/microsoft/vcpkg/issues/19353)

2. [Unresolved \_\_\_stdio\_common\_vsprintf\_s, what library has this?](https://stackoverflow.com/questions/32740172/unresolved-stdio-common-vsprintf-s-what-library-has-this)

3. [[mpfr] Build error · Issue #23641 · microsoft/vcpkg · GitHub](https://github.com/microsoft/vcpkg/issues/23641)

4. <https://github.com/Microsoft/vcpkg/issues>

# User

After cloning using:

# Bing

It

# User

For ISL building after cloning:

git clone git://repo.or.cz/isl.git

When I use git pull

it gives me error in msys2:

$ git pull

fatal: not a git repository (or any of the parent directories): .git

# Bing

The error message fatal: not a git repository (or any of the parent directories): .git typically means that you're trying to run a Git command outside of a Git repository[**1**](https://stackoverflow.com/questions/11961600/fatal-not-a-git-repository-or-any-of-the-parent-directories-from-git-status). Here are a few things you can check:

1. **Check your current directory**: Make sure you're in the correct directory where the Git repository (in this case, the isl repository you cloned) is located[**1**](https://stackoverflow.com/questions/11961600/fatal-not-a-git-repository-or-any-of-the-parent-directories-from-git-status). You can use the pwd command to print the current directory and confirm you're in the right place.
2. **Check for a `.git` directory**: A Git repository typically has a .git directory at its root where Git stores all the repository data[**4**](https://www.datree.io/resources/git-error-fatal-not-a-git-repository). You can use the ls -la command to list all files and directories, including hidden ones, and check if the .git directory is present[**4**](https://www.datree.io/resources/git-error-fatal-not-a-git-repository).
3. **Re-clone the repository**: If you're in the correct directory and the .git directory is missing, the repository might have been corrupted or not fully cloned. In this case, you might need to re-clone the repository[**1**](https://stackoverflow.com/questions/11961600/fatal-not-a-git-repository-or-any-of-the-parent-directories-from-git-status).

If you're still having trouble, please let me know! I'm here to help.

Source(s)

1. ["fatal: Not a git repository (or any of the parent directories)" from ...](https://stackoverflow.com/questions/11961600/fatal-not-a-git-repository-or-any-of-the-parent-directories-from-git-status)

2. [Fatal: not a git repository (or any of the parent directories)](https://careerkarma.com/blog/git-fatal-not-a-git-repository/)

3. [Git pull fails on a worktree: Not a git repository (or any of the ...](https://stackoverflow.com/questions/28524515/git-pull-fails-on-a-worktree-not-a-git-repository-or-any-of-the-parent-directo)

4. [Git error - Fatal: Not a git repository and how to fix it | Datree.io](https://www.datree.io/resources/git-error-fatal-not-a-git-repository)

5. [How To Fix "fatal: not a git repository" in Git? - Tim Mouskhelichvili](https://timmousk.com/blog/fatal-not-a-git-repository/)

# User

After cloning isl and using git pull, as per instructions I ran ./autogen.sh in msys2 shell. It gave error:

$ ./autogen.sh
./autogen.sh: line 2: autoreconf: command not found

Where is that located? Can you search for this error?

# Bing

The error autoreconf: command not found indicates that the autoreconf tool is not available in your current environment[**1**](https://github.com/msys2/MINGW-packages/issues/1010). autoreconf is part of the GNU Autotools, a suite of programming tools designed to assist in making source code packages portable across many Unix-like systems[**1**](https://github.com/msys2/MINGW-packages/issues/1010).

In MSYS2, you can install autoreconf by installing the autoconf package with the package manager pacman[**2**](https://stackoverflow.com/questions/72019820/how-to-fix-configure-in-msys2)[**4**](https://www.programmerhat.com/autoreconf-command-not-found/). Here's how you can do it:

pacman -Syu --noconfirm

pacman -S --noconfirm autoconf autoconf-archive automake make libtool pkg-config

The first command updates the package manager information[**2**](https://stackoverflow.com/questions/72019820/how-to-fix-configure-in-msys2). The second command installs the packages you need, including autoconf which provides the autoreconf tool[**2**](https://stackoverflow.com/questions/72019820/how-to-fix-configure-in-msys2).

After running these commands, you should be able to run autoreconf in your MSYS2 shell[**2**](https://stackoverflow.com/questions/72019820/how-to-fix-configure-in-msys2).

Let me know if you need further assistance!

Source(s)

1. [autoreconf command not found · Issue #1010 · msys2/MINGW-packages](https://github.com/msys2/MINGW-packages/issues/1010)

2. [windows - How to fix ./configure in MSYS2? - Stack Overflow](https://stackoverflow.com/questions/72019820/how-to-fix-configure-in-msys2)

3. [Autotools - MSYS2](https://www.msys2.org/docs/autotools/)

4. [How to fix “autoreconf: command not found” in Linux](https://www.programmerhat.com/autoreconf-command-not-found/)

# User

I want to know with which paramters clang was installed in Msys2 shell. For gcc, I can view that using gcc -v which lists down ../configure followed by all parameters. What si equivalent of that for clang?

# Bing

In the MSYS2 environment, the installation of Clang doesn't typically involve manual configuration with parameters like GCC. Instead, it's installed via the package manager pacman[**4**](https://packages.msys2.org/package/mingw-w64-x86_64-clang).

However, if you're looking to check the version of Clang and its various features, you can use the command clang --version in your MSYS2 shell. This will provide information about the installed version of Clang, but it won't provide the exact configuration parameters used during its installation.

If you're interested in using Clang as your compiler instead of GCC in a CMake project, you can specify it with the -DCMAKE\_C\_COMPILER=clang -DCMAKE\_CXX\_COMPILER=clang++ parameters[**5**](https://stackoverflow.com/questions/40061101/how-to-access-windows-enviroment-variables-on-msys).

Please note that the exact parameters used during the installation of Clang in the MSYS2 environment might not be directly accessible or visible as it is with GCC's ../configure parameters. The installation and configuration of Clang are handled by the package manager and the specifics can vary based on the package details. If you need more specific information, you might need to check the package documentation or source files.

Source(s)

1. [Group: mingw-w64-clang-x86\_64-toolchain - MSYS2 Packages](https://packages.msys2.org/groups/mingw-w64-clang-x86_64-toolchain)

2. [TIL: Ninja & MSYS2 Clang - DevLog](https://duerrenberger.dev/blog/2021/05/06/til-ninja-msys2-clang/)

3. [Getting Started with Clang and Visual Studio Code on Windows with MSYS2 ...](https://solarianprogrammer.com/2021/06/11/install-clang-windows-msys2-mingw-w64/)

4. [Package: mingw-w64-x86\_64-clang - MSYS2 Packages](https://packages.msys2.org/package/mingw-w64-x86_64-clang)

5. [How to access windows enviroment variables on MSYS?](https://stackoverflow.com/questions/40061101/how-to-access-windows-enviroment-variables-on-msys)

6. <https://llvm.org/>

7. <https://www.msys2.org/>

8. <https://clang.llvm.org/>

9. <https://mirror.msys2.org/mingw/mingw64/mingw-w64-x86_64-clang-17.0.6-7-any.pkg.tar.zst>