# COMPILING WRF & WPS

Kelly Werner, NCAR/MMM





1. Check system requirements

2. Install libraries

3. Obtain source code

4. Compile WRF

5. Compile WPS

### WRF/WPS COMPILING PAGE





#### PAGE CONTENTS

Compiling WRF and WPS	
System Environment Tests	+
Install Libraries	+
Library Compatibility Tests	+
Compiling WRF	
Compiling WPS	
Troubleshooting	+
Next Steps	

Start / Compiling WRF and WPS

### **Compiling WRF and WPS**

\$

This page provides guidance for compiling WRF and WPS using either a GNU/gfortran/gcc compiler or an intel compiler. **Follow the** steps in order for a successful compile.

#### **A** Important

WRF User Support staff cannot help with errors that occur during system environment tests or during the process of installing libraries. If such errors should arise, please contact a systems administrator at your institution to assist. You may also find it useful to do a web search for the errors.

#### **System Environment Tests**

https://www2.mmm.ucar.edu/wrf/wrf\_tutorial/build/html/compiling\_tutorial/compiling\_tutorial.html

### Mandatory Requirements

- Fortran compiler (e.g., gfortran)
- C compiler/cpp
- gcc

# CHECK SYSTEM Requirements

### Check if they exist on your system

> which gfortran
> which cpp
> which gcc
If installed, you will be given a path for each

**NOTE:** You cannot compile newer versions of WRF with older compilers, or vice versa. Check the version:

> gcc --version

#### Test #1 - Fixed Format Fortran Test

Issue the following two commands from within the *tests* directory.

gfortran TEST\_1\_fortran\_only\_fixed.f

a.out

If successful, the following message should print to the screen:

SUCCESS test 1 fortran only fixed format

#### Test #2: Free Format Fortran

Issue the following two commands from within the tests directory.

gfortran TEST\_2\_fortran\_only\_free.f90

./a.out

If successful, the following message should print to the screen:

Assume Fortran 2003: has FLUSH, ALLOCATABLE, derived type, and ISO C Binding SUCCESS test 2 fortran only free format

#### Test #3: C

Issue the following two commands from within the *tests* directory.

gcc TEST\_3\_c\_only.c

./a.out

If successful, the following message should print to the screen:

SUCCESS test 3 c only

#### Test #4: Fortran Calling a C Function

gcc and gfortran have different defaults, so they are forced to both always use 64 bit [-m64] when combining them. Issue the following two commands from within the *tests* directory.

gcc -c -m64 TEST\_4\_fortran+c\_c.c

gfortran -c -m64 TEST\_4\_fortran+c\_f.f90

gfortran -m64 TEST\_4\_fortran+c\_f.o TEST\_4\_fortran+c\_c.o

./a.out

...................................

Tests are available for checking that your fortran compiler is properly built, and is compatible with the C compiler.

1. Check system requirements

2. Install libraries

3. Obtain source code

4. Compile WRF

5. Compile WPS

### NetCDF (for WRF & WPS)

- NetCDF (v3 or v4)
- NetCDF-c and netCDF-fortran
- If using netCDF-4 capabilities, you must install HDF5 before compiling netCDF
  - See <u>Notes on Building NetCDF4 for WRF</u>

### Libraries for GRIB2 Meteorological Data

- Only required for WPS versions prior to v4.4
- JasPer (JPEG 2000 "lossy" compression library)
- PNG ("lossless" compression library)
- Zlib (compression library used by PNG)

### "Optional" Libraries

- MPI: e.g., MPICH or OpenMPI if planning to run with multiple processors
- HDF5 if needing to use file compression

### Required Libraries

#### Environment Settings for a gfortran/GNU Compiler

export dir=/full-path-to-libs-directory/libs export dir=/full-path-to-libs-directory/libs export PATH=.:/\$dir/netcdf/bin:/\$dir/bin:\${PATH} export LD\_LIBRARY\_PATH=/\$dir/lib:/\$dir/netcdf/lib:/\$dir/grib2/lib export JASPERLIB=/\$dir/grib2/lib export JASPERINC=/\$dir/grib2/include export NETCDF=/\$dir/netcdf export CC=gcc export CXX=g++ export FC=gfortran export FCFLAGS=-m64 export FFLAGS=-m64 export LDFLAGS=-L\$dir/grib2/lib export CPPFLAGS=-I\$dir/grib2/include

#### **Environment Settings for an Intel Compiler**

export dir=/full-path-to-libs-directory/libs export PATH=.:/\$dir/netcdf/bin:/\$dir/bin:\${PATH} export LD\_LIBRARY\_PATH=/\$dir/lib:/\$dir/netcdf/lib:/\$dir/grib2/lib export JASPERLIB=/\$dir/grib2/lib export JASPERINC=/\$dir/grib2/include export NETCDF=/\$dir/netcdf export CC=icc export CXX=icpc export CFLAGS='-03 -xHost -ip -no-prec-div -static-intel' export CXXFLAGS='-03 -xHost -ip -no-prec-div -static-intel' export F77=ifort export FC=ifort export F90=ifort export FFLAGS='-03 -xHost -ip -no-prec-div -static-intel' export CPP='icc -E' export CXXCPP='icpc -E' export LDFLAGS=-L\$dir/grib2/lib export CPPFLAGS=-I\$dir/grib2/include

### Set Paths in Environment Script

#### Environment Scripts:

- e.g., .cshrc, .bash, .tcshrc
- Found in your home directory
- Issue "Is -a" to see files with a "." in front of them



#### Ensures settings are in place every time you open a new terminal window!

# Installing MPI

#### Install mpich

If planning to use more than a single processor when running WRF, it is necessary to install an MPI library. From inside the *libs* directory, issue the following commands:

wget https://www2.mmm.ucar.edu/wrf/OnLineTutorial/compile_tutorial/tar_files/mpich-3.0.4.tar.gz	
tar -xf mpich-3.0.4.tar.gz	
cd mpich-3.0.4	
./configureprefix=\$dir	
make 2>&1	
make install	
cd	
rm -rf mpich*	

Step-by-step instructions for installing remaining libraries (zlib, HDF5, netCDF-c, netCDF-fortran libpng, jasper) available from <u>Compiling WRF and WPS</u>

### Library Compatibility Checks

# Make sure libraries are compatible with compilers

#### <u>Test 1</u>

Fortran + C + netCDF

#### <u>Test 2</u>

Fortran + C + netCDF + MPI

#### Test #1: Fortran + C + netCDF

This test requires the include file from the netCDF package be in this directory. From the tests directory, copy the file "here":

#### cp \${NETCDF}/include/netcdf.inc .

Compile the Fortran and C codes for the purpose of this test (the -c option specifies no executable is to be built). Issue the following commands:

gfortran -c 01\_fortran+c+netcdf\_f.f

gcc -c 01\_fortran+c+netcdf\_c.c

gfortran 01\_fortran+c+netcdf\_f.o 01\_fortran+c+netcdf\_c.o -L\${NETCDF}/lib -lnetcdff -lnetcdf

./a.out

If successful, the following message should print to the screen:

C function called by Fortran Values are xx = 2.00 and ii = 1 SUCCESS test 1 fortran + c + netcdf

#### Test #2: Fortran + C + netCDF + MPI

From the tests directory, issue the following commands:

mpif90 -c 02\_fortran+c+netcdf+mpi\_f.f

mpicc -c 02\_fortran+c+netcdf+mpi\_c.c

mpif90 02\_fortran+c+netcdf+mpi\_f.o 02\_fortran+c+netcdf+mpi\_c.o -L\${NETCDF}/lib -lnetcdff -lnet mpirun ./a.out

If successful, the following message should print to the screen:

C function called by Fortran Values are xx = 2.00 and ii = 1 status = 2 SUCCESS test 2 fortran + c + netcdf + mpi

1. Check system requirements

2. Install libraries

3. Obtain source code

4. Compile WRF

5. Compile WPS

#### WRF SOURCE CODE REGISTRATION AND DOWNLOAD

Beginning with V4.0 of the WRF/WRFDA/WRF-Chem/WPS code, all release downloads and corresponding information will be available from our public WRF-Model GitHub page. For code downloads prior to V4.0, click here.

There are 2 methods to obtain the WRF-Modeling System source code:

1. The recommended method is to clone the code from our public GitHub repository. This can be done in the command-line. This options requires an installation of git (which most modern systems likely already have – you can check with the command (csh e.g.): which git). This method provides more flexibility to update the version and facilitates the most direct method for contributing development back into the WRF-Model code base.

WRF Model Source Code (includes WRF, WRFDA, & WRF-Chem): git close https://github.com/urf-model/WPF

git clone https://github.com/wrf-model/WRF

WRF Preprocessing System Source Code :
git clone https://github.com/wrf-model/WPS

See the archives page for all release notes.

Since V4.0, WRFDA/WRFPlus code is now fully-integrated into the WRF code. See the <u>WRFDA V4.0 Update Summary</u> and chapter 6 of the <u>Users Guide</u> for additional information.

2. The second method is to aquire the code through the archive file on GitHub. The disadvantage to this method is the lack of flexibility with the ability to troubleshoot with version control. Archive files are provided in both zip and tar.gz formats. Each release provides an archive file, and users should download the archive file for the most relevant released version.

WRF Model Archive File (includes WRF, WRFDA, WRF-Chem)

WRF Preprocessing System (WPS) Model Archive File

# Obtain WRF/WPS Source Code

#### Obtain Source Code from:

#### WRF Source Codes and Graphics Software

- 1. Choose 'New User,' and then register, or
- 2. Click 'Returning User,' enter your email, and go to the download page.

# All code is stored in a GitHub repository, and can be obtained by:

- Cloning from GitHub
- Downloading archived tar file from GitHub

\*\*Must have 'git' installed on your system!

### Cloning Source Code from GitHub

#### Clone WRF from GitHub repository "wrf-model"

cheyenne:/glade/scratch/kkeene>git clone --recurse-submodule https://git@github.com/wrf-model/WRF Cloning into 'WRF'... remote: Enumerating objects: 62656, done. remote: Counting objects: 100% (250/250), done. remote: Compressing objects: 100% (134/134), done. remote: Total 62656 (delta 130), reused 189 (delta 116), pack-reused 62406 Receiving objects: 100% (62656/62656), 266.71 MiB | 22.74 MiB/s, done. Resolving deltas: 100% (48594/48594), done. Updating files: 100% (4758/4758), done. Submodule 'phys/noahmp' (https://github.com/NCAR/noahmp) registered for path 'phys/noahmp' Cloning into '/glade/scratch/kkeene/WRF/phys/noahmp'... remote: Enumerating objects: 1149, done. remote: Counting objects: 100% (285/285), done. remote: Compressing objects: 100% (216/216), done. remote: Total 1149 (delta 76), reused 245 (delta 62), pack-reused 864 Receiving objects: 100% (1149/1149), 7.23 MiB | 21.72 MiB/s, done. Resolving deltas: 100% (377/377), done. Submodule path 'phys/noahmp': checked out '3be0b2860dab167006a0b3c4822e234ca253c3df' chavannas /alada/caratch/kkaanas

> Clone WPS: > git clone https://github.com/wrf-model/WPS

1. Check system requirements

2. Install libraries

3. Obtain source code

4. Compile WRF

5. Compile WPS

# Step 1: Configure for WRF

#### In the WRF directory, issue:

./configure

#### Configuration Output

configure.wrf

\$JASPERLIB or \$JASPERINC not found in environment, configuring to build without grib2 I/0...

Please select from among the following Linux x86\_64 options:

(serial)	2. (smpar)	3. (dmpar)	4. (dm+sm)	PGI (pgf90/gcc)
(serial)	6. (smpar)	7. (dmpar)	8. (dm+sm)	PGI (pgf90/pgcc): SGI MPT
(serial)	10. (smpar)	11. (dmpar)	12. (dm+sm)	PGI (pgf90/gcc): PGI accelerator
(serial)	14. (smpar)	15. (dmpar)	16. (dm+sm)	INTEL (ifort/icc)
			17. (dm+sm)	INTEL (ifort/icc): Xeon Phi (MIC architecture)
(serial)	19. (smpar)	20. (dmpar)	21. (dm+sm)	INTEL (ifort/icc): Xeon (SNB with AVX mods)
(serial)	23. (smpar)	24. (dmpar)	25. (dm+sm)	INTEL (ifort/icc): SGI MPT
(serial)	27. (smpar)	28. (dmpar)	29. (dm+sm)	INTEL (ifort/icc): IBM POE
(serial)		31. (dmpar)		PATHSCALE (pathf90/pathcc)
(serial)	33. (smpar)	34. (dmpar)	35. (dm+sm)	GNU (gfortran/gcc)
(serial)	37. (smpar)	38. (dmpar)	39. (dm+sm)	IBM (xlf90_r/cc_r)
(serial)	41. (smpar)	42. (dmpar)	43. (dm+sm)	PGI (ftn/gcc): Cray XC CLE
(serial)	45. (smpar)	46. (dmpar)	47. (dm+sm)	CRAY CCE (ftn \$(NOOMP)/cc): Cray XE and XC
(serial)	49. (smpar)	50. (dmpar)	51. (dm+sm)	INTEL (ftn/icc): Cray XC
(serial)	53. (smpar)	54. (dmpar)	55. (dm+sm)	PGI (pgf90/pgcc)
(serial)	57. (smpar)	58. (dmpar)	59. (dm+sm)	PGI (pgf90/gcc): -f90=pgf90
(serial)	61. (smpar)	62. (dmpar)	63. (dm+sm)	PGI (pgf90/pgcc): -f90=pgf90
(serial)	65. (smpar)	66. (dmpar)	67. (dm+sm)	INTEL (ifort/icc): HSW/BDW
(serial)	69. (smpar)	70. (dmpar)	71. (dm+sm)	INTEL (ifort/icc): KNL MIC
(serial)	73. (smpar)	74. (dmpar)	75. (dm+sm)	FUJITSU (frtpx/fccpx): FX10/FX100 SPARC64 IXfx/Xlf>
	<pre>(serial) (serial) (serial)</pre>	<pre>(serial) 2. (smpar) (serial) 6. (smpar) (serial) 10. (smpar) (serial) 14. (smpar) (serial) 14. (smpar) (serial) 23. (smpar) (serial) 27. (smpar) (serial) 33. (smpar) (serial) 37. (smpar) (serial) 41. (smpar) (serial) 45. (smpar) (serial) 53. (smpar) (serial) 57. (smpar) (serial) 61. (smpar) (serial) 65. (smpar) (serial) 69. (smpar) (serial) 73. (smpar)</pre>	<pre>(serial) 2. (smpar) 3. (dmpar) (serial) 6. (smpar) 7. (dmpar) (serial) 10. (smpar) 11. (dmpar) (serial) 14. (smpar) 15. (dmpar) (serial) 14. (smpar) 15. (dmpar) (serial) 23. (smpar) 24. (dmpar) (serial) 27. (smpar) 28. (dmpar) (serial) 27. (smpar) 28. (dmpar) (serial) 33. (smpar) 34. (dmpar) (serial) 37. (smpar) 38. (dmpar) (serial) 37. (smpar) 38. (dmpar) (serial) 41. (smpar) 42. (dmpar) (serial) 45. (smpar) 50. (dmpar) (serial) 53. (smpar) 54. (dmpar) (serial) 57. (smpar) 58. (dmpar) (serial) 61. (smpar) 62. (dmpar) (serial) 65. (smpar) 70. (dmpar) (serial) 69. (smpar) 74. (dmpar)</pre>	<pre>(serial) 2. (smpar) 3. (dmpar) 4. (dm+sm) (serial) 6. (smpar) 7. (dmpar) 8. (dm+sm) (serial) 10. (smpar) 11. (dmpar) 12. (dm+sm) (serial) 14. (smpar) 15. (dmpar) 16. (dm+sm) (serial) 14. (smpar) 20. (dmpar) 21. (dm+sm) (serial) 23. (smpar) 24. (dmpar) 25. (dm+sm) (serial) 27. (smpar) 28. (dmpar) 29. (dm+sm) (serial) 33. (smpar) 34. (dmpar) 35. (dm+sm) (serial) 37. (smpar) 38. (dmpar) 35. (dm+sm) (serial) 37. (smpar) 38. (dmpar) 39. (dm+sm) (serial) 41. (smpar) 42. (dmpar) 43. (dm+sm) (serial) 45. (smpar) 50. (dmpar) 51. (dm+sm) (serial) 53. (smpar) 54. (dmpar) 55. (dm+sm) (serial) 57. (smpar) 58. (dmpar) 59. (dm+sm) (serial) 61. (smpar) 62. (dmpar) 63. (dm+sm) (serial) 65. (smpar) 66. (dmpar) 71. (dm+sm) (serial) 69. (smpar) 70. (dmpar) 71. (dm+sm)</pre>

Enter selection [1-75] : 34

Compile for nesting? (1=basic, 2=preset moves, 3=vortex following) [default 1]:

### Additional WRF Configuration Options

### > ./configure -d

- No optimization
- Extra debugging

#### > ./configure -D

- No optimization
- Checks uninitialized variables

### > ./configure -r8

- Double-precision
- Works for GNU, Intel, and PGI compilers

Step 2: Compile WRF



\*\*Compilation will take ~10-50 mins\*\*

### Compiling With Multiple Processors

To build WRF with multiple processors, add the "J" variable to the compile command: ./compile em\_real -j 6 >& compile.log

# of Processors	Time to Compile
1	38 Mins 43 Secs
2	25 Mins 4 Secs
3	21 Mins 48 Secs
4	20 Mins 14 Secs
5	19 Mins 16 Secs
6	19 Mins 10 Secs

\*Compiled with GNU V10.1.0

# Successful Compile

ideal.exe executable, but with the same name

Real data case

# If compiling is successful, this message appears at the end of the compile log:

> Executables successfully built <	wrf.exe – model executable real.exe – real data initialization
-rwxr-xr-x 1 wrfhelp ncar 54128088 Jul 22 15:54 main/ndown.exe -rwxr-xr-x 1 wrfhelp ncar 54180032 Jul 22 15:54 main/real.exe -rwxr-xr-x 1 wrfhelp ncar 53464304 Jul 22 15:54 main/tc.exe -rwxr-xr-x 1 wrfhelp ncar 60686816 Jul 22 15:52 main/wrf.exe	<b>ndown.exe</b> – separate one-way nesting <b>tc.exe</b> – for tropical cyclone bogusing

These executables are linked to 2 different directories. You can run a real-data case in either directory. Idealized cases must be run in the appropriate em\_*case* directory.

- WRF/run
- WRF/test/em\_case

# Choosing the Right Compiler

### <u>Compile</u> WRF V4.5.2 dmpar/basic nesting 6 processors

Compiler	Compile Time	Run Time
GNU/gcc 12.2.0	25 Mins	3.92 Mins
Intel 2023.2.1	37 Mins	2.20 Min

### <u>Run</u>

Single domain Small domain (75x70), coarse (30km) resolution 12 hours 8 processors

1. Check system requirements

2. Install libraries

3. Obtain source code

4. Compile WRF

5. Compile WPS

### Step 1: Configure for WPS

#### Inside the WPS/ directory, type

- > export WRF\_DIR=path-to-top-level-WRF/WRF
- > ./configure

### Configuration Output

configure.wps

Will use NETCDF in dir: /glade/u/apps/derecho/23.09/spack/opt/spack/netcdf/4.9.2/gcc/13.2.0/zywl Using WRF I/O library in WRF build identified by \$WRF\_DIR: ../wrf \$JASPERLIB or \$JASPERINC not found in environment. Using default values for library paths...

Please select from among the following supported platforms.

1.	Linux	x86_64,	gfortran	(serial)		
2.	Linux	x86_64,	gfortran	(serial_NO_G	RIB2)	
з.	Linux	x86_64,	gfortran	(dmpar)		
4.	Linux	x86_64,	gfortran	(dmpar_NO_GF	RIB2)	
5.	Linux	x86_64,	PGI compiler	(serial)		
6.	Linux	x86 64,	PGI compiler	c (serial_N	IO_GRIB2)	
7.	Linux	x86_64,	PGI compiler	c (dmpar)	-	
8.	Linux	x86_64,	PGI compiler	c (dmpar_NC	GRIB2)	
9.	Linux	x86_64,	PGI compiler	, SGI MPT	(serial)	
10.	Linux	x86_64,	PGI compiler	, SGI MPT	(serial_NO_	GRIB2)
11.	Linux	x86_64,	PGI compiler	, SGI MPT	(dmpar)	
12.	Linux	x86_64,	PGI compiler	, SGI MPT	(dmpar_NO_G	RIB2)
13.	Linux	x86_64,	IA64 and Opt	eron (ser	ial)	
14.	Linux	x86_64,	IA64 and Opt	eron (ser	ial_NO_GRIB	2)
15.	Linux	x86_64,	IA64 and Opt	eron (dmp	par)	
16.	Linux	x86 64,	IA64 and Opt	eron (dmp	ar_NO_GRIB2	)
17.	Linux	x86_64,	Intel oneAPI	[ compilers	(serial)	
18.	Linux	x86_64,	Intel oneAPI	[ compilers	(serial_N	O_GRIB2)
19.	Linux	x86_64,	Intel oneAPI	[ compilers	(dmpar)	
20.	Linux	x86_64,	Intel oneAPI	[ compilers	(dmpar_NO	GRIB2)
21.	Linux	x86_64,	Intel Classi	c compilers	(serial)	
22.	Linux	x86_64,	Intel Classi	.c compilers	(serial_	NO_GRIB2)
23.	Linux	x86_64,	Intel Classi	.c compilers	(dmpar)	
24.	Linux	x86_64,	Intel Classi	.c compilers	(dmpar_N	O_GRIB2)
25.	Linux	x86_64,	Intel Classi	c compilers,	SGI MPT	(serial)
26.	Linux	x86_64,	Intel Classi	c compilers,	SGI MPT	(serial_NO_GRIB2)
27.	Linux	x86_64,	Intel Classi	.c compilers,	SGI MPT	(dmpar)
28.	Linux	x86_64,	Intel Classi	.c compilers,	SGI MPT	(dmpar_NO_GRIB2)
29.	Linux	x86_64,	Intel Classi	.c compilers,	IBM POE	(serial)
30.	Linux	x86_64,	Intel Classi	c compilers,	IBM POE	(serial_NO_GRIB2)
31.	Linux	x86_64,	Intel Classi	c compilers,	IBM POE	(dmpar)
32.	Linux	x8664,	Intel Classi	c compilers,	IBM POE	(dmpar NO GRIB2)

Choose a serial compile for WPS (even if WRF is compiled with a parallel option)

- \* Exception: You are using a VERY large domain (1000's x 1000's)
- NOTE: if you do compile WPS in parallel, ungrib.exe must run serially

# Configure WPS to Use Internal Compression Libraries

#### Feature available in WPSv4.4+

./configure --build-grib2-libs

Installs these compressions libraries for use with GRIB2 input

- zlib
- libpng
- JasPer

### Step 2: Compile WPS

### In the WPS/ directory, type

```
./compile >& log.compile
 *Compilation should be quick.
```

If successful, these executables should be in your WPS/ directory (linked from their source code directories):

```
geogrid.exe -> geogrid/src/geogrid.exe
ungrib.exe -> ungrib/src/ungrib.exe
metgrid.exe -> metgrid/src/metgrid.exe
```

1. Check system requirements

2. Install libraries

3. Obtain source code

4. Compile WRF

5. Compile WPS

### Search for errors in the *compile.log*

- Search for 'Error' with a capital 'E'
- Typically the first 'Error' in the file is the culprit

# Failed WRF Compile

### Visit the <u>WRF & MPAS-A Support Forum</u>

- See <u>Frequently Asked Questions</u> (FAQ)
- Search the forum to see if your issue has already been addressed if not, post a new topic

### Before Recompiling

- Issue the ./clean –a command
- Reconfigure
  - If you need to make changes to the *configure.wrf* file, do that after issuing *./configure*, and then save the edited file
- Recompile



Failed WPS Compile: No geogrid or metgrid



### Did WRF compile successfully?

geogrid and metgrid use the external I/O libraries from WRF, which are built when WRF is installed



Check that you are using the same compiler (& version) that was used to compile WRF



Check that you are using the same version of netCDF that was used to compile WRF

# Failed WPS Compile: No ungrib



Make sure jasper, zlib, and libpng libraries are correctly installed



Make sure you are using the correct path format for these lines in *configure.wps* 

COMPRESSION\_LIBS = -L/\${DIR}/UNGRIB\_LIBRARIES/lib -ljasper -lpng -lz COMPRESSION\_INC = -I/\${DIR}/UNGRIB\_LIBRARIES/include



Alternatively, use the option to build these libraries internally

Using "clean -a"

- The './clean –a' command should be used when modifications have been made to the configure.wrf(wps) file, or any changes to the registry. If so, issue 'clean –a' prior to recompiling.
- Modifications to subroutines within the code will require a recompile, but DO NOT require a 'clean –a', nor a reconfigure. Simply recompile. This compilation should be much faster than a clean compile.



