2013 Joint DTC-EMC-JCSDA GSI Tutorial, NCWCP College Park, MD

GSI Fundamentals (1): Setup and Compilation

Donald Stark

National Center for Atmospheric Research (NCAR)

The Developmental Testbed Center (DTC)

Monday 5 August, 2013

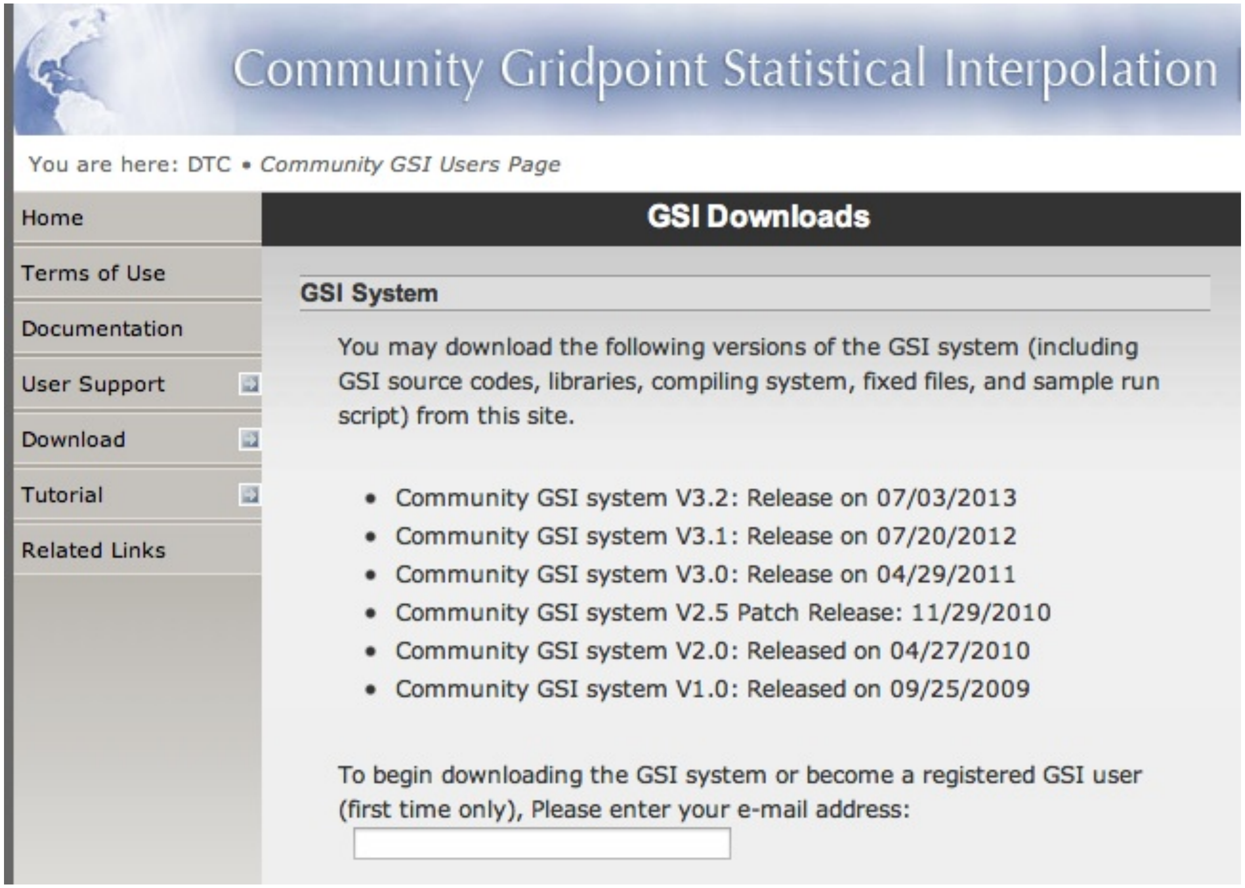
Outline

- GSI fundamentals (1): Setup and Compilation
  - Where to get the code
  - Directory structure
  - Unpacking, setup, & build
  - Porting build to new platforms
- GSI fundamentals (2): Run and Namelist
- GSI fundamentals (3): Diagnostics
- GSI fundamentals (4): Applications

This talk is tailored based on Chapter 2 of the GSI User's Guide for Community Release V3.2

Downloading the Source Code

- All of the GSI source code can be obtained from:
  - http://www.dtcenter.org/com-GSI/users/downloads/index.php


Community Gridpoint Statistical Interpolation
You are here: DTC • Community GSI Users Page
Home
Terms of Use
Documentation
User Support
Download
Tutorial
Related Links

GSI Downloads

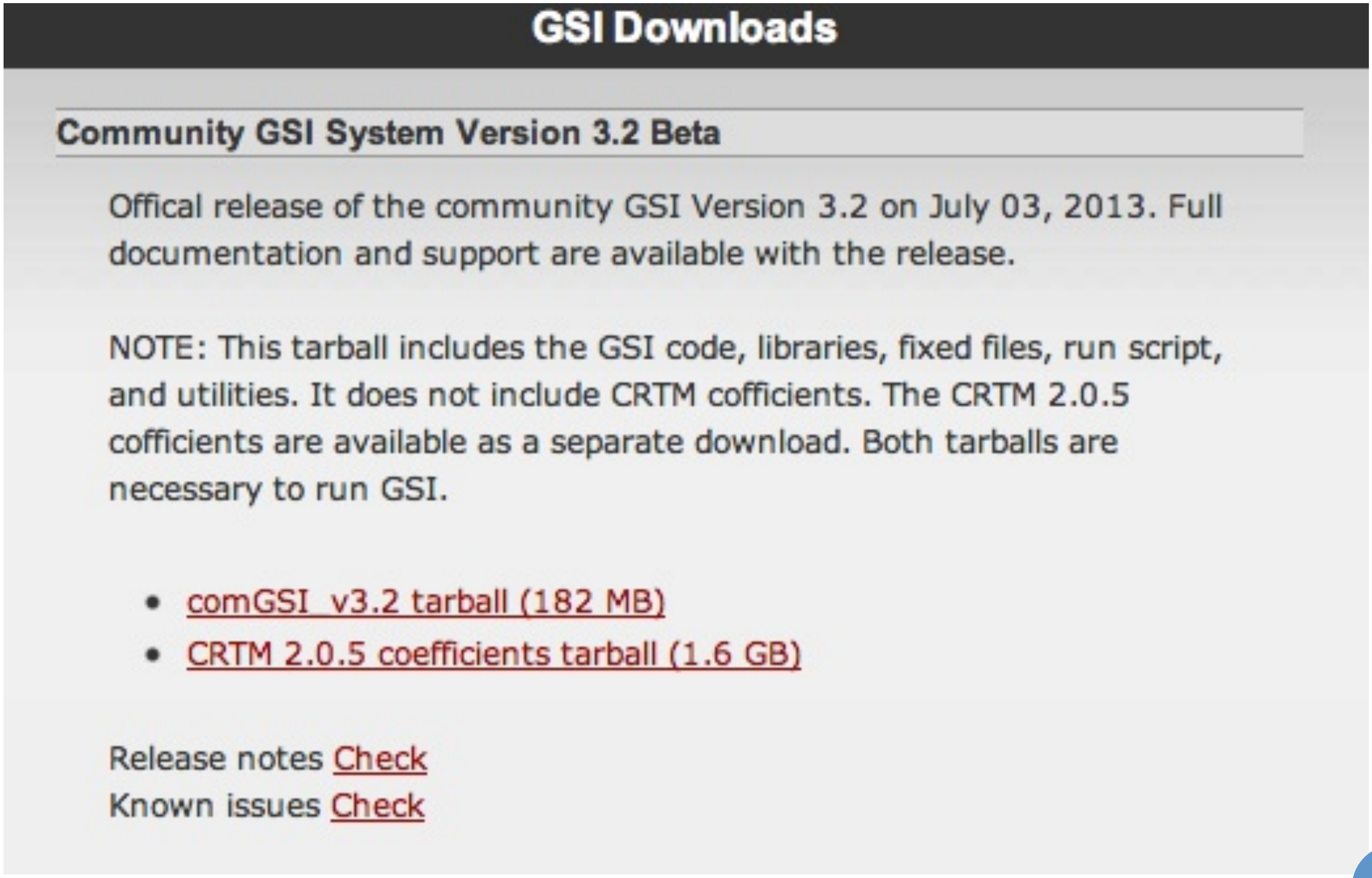
GSI System

You may download the following versions of the GSI system (including GSI source codes, libraries, compiling system, fixed files, and sample run script) from this site.

- Community GSI system V3.2: Release on 07/03/2013
- Community GSI system V3.1: Release on 07/20/2012
- Community GSI system V3.0: Release on 04/29/2011
- Community GSI system V2.5 Patch Release: 11/29/2010
- Community GSI system V2.0: Released on 04/27/2010
- Community GSI system V1.0: Released on 09/25/2009

To begin downloading the GSI system or become a registered GSI user (first time only), Please enter your e-mail address:

Downloading Source code



GSI Downloads

Community GSI System Version 3.2 Beta

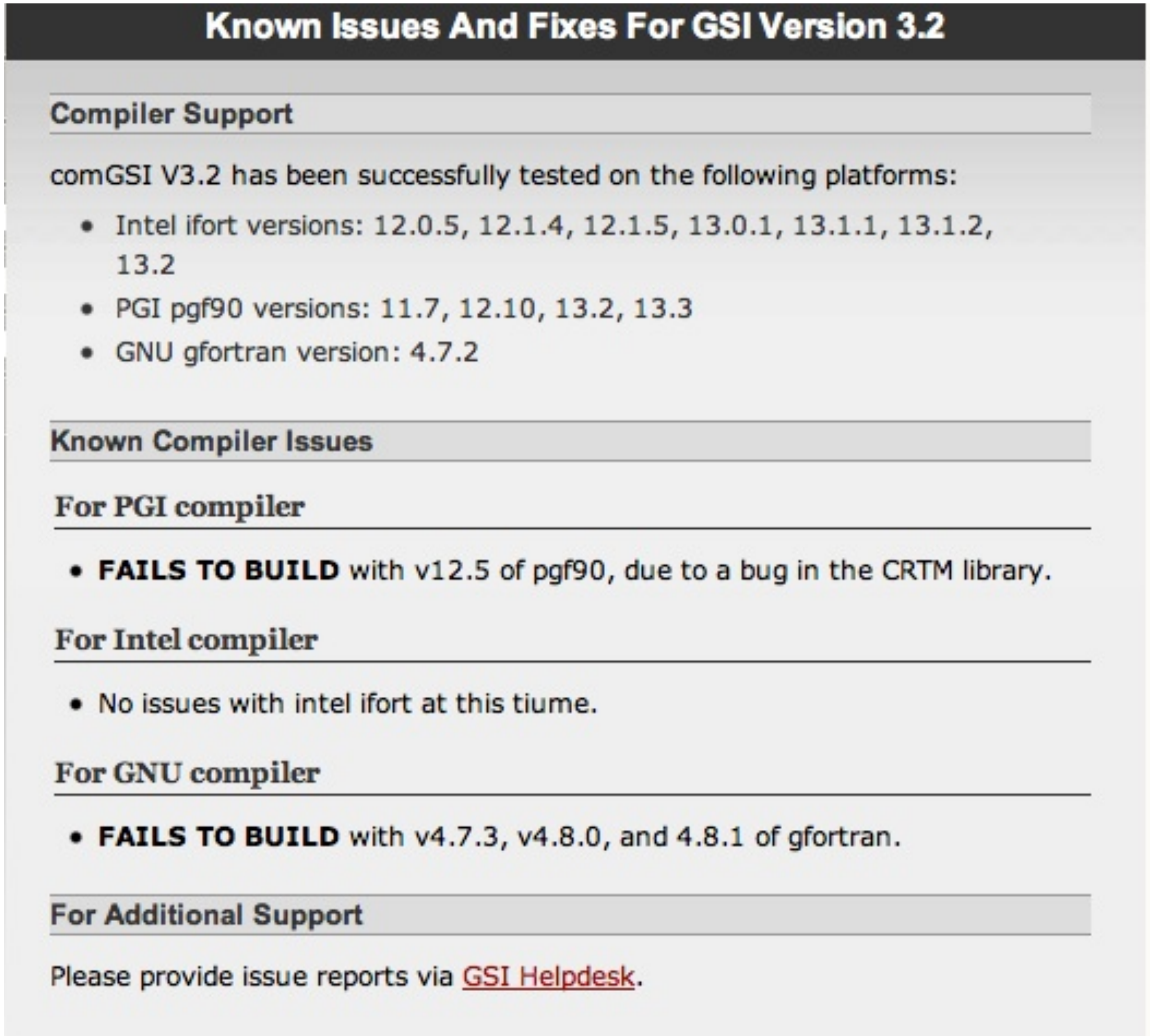
Offical release of the community GSI Version 3.2 on July 03, 2013. Full documentation and support are available with the release.

NOTE: This tarball includes the GSI code, libraries, fixed files, run script, and utilities. It does not include CRTM cofficients. The CRTM 2.0.5 cofficients are available as a separate download. Both tarballs are necessary to run GSI.

- comGSI_v3.2 tarball (182 MB)
- CRTM 2.0.5 coefficients tarball (1.6 GB)

Release notes Check
Known issues Check

Known Issues



Known Issues And Fixes For GSI Version 3.2

Compiler Support

comGSI V3.2 has been successfully tested on the following platforms:

- Intel ifort versions: 12.0.5, 12.1.4, 12.1.5, 13.0.1, 13.1.1, 13.1.2, 13.2
- PGI pgf90 versions: 11.7, 12.10, 13.2, 13.3
- GNU gfortran version: 4.7.2

Known Compiler Issues

For PGI compiler

- **FAILS TO BUILD** with v12.5 of pgf90, due to a bug in the CRTM library.

For Intel compiler

- No issues with intel ifort at this tiume.

For GNU compiler

- **FAILS TO BUILD** with v4.7.3, v4.8.0, and 4.8.1 of gfortran.

For Additional Support

Please provide issue reports via GSI Helpdesk.

Supported Platforms

Platform	F90 compiler	C compiler
IBM*	xlf	xlc
Linux	Intel (ifort)	Intel (icc)
	Intel (ifort)	Gnu (gcc)
	PGI (pgf90)	PGI (pgcc)
	PGI (pgf90)	Gnu (gcc)
	Gnu (gfortran)	Gnu (gcc)
Mac*	PGI (pgf90)	PGI (pgcc)

* Legacy support provided on platforms no longer available for testing

Unpack Downloads

- Two tar files
  - comGSI_v3.2.tar.gz
  - CRTM_Coefficients-2.0.5.tar.gz
- Unpack source code & CRTM coefficients
  - gunzip *.tar.gz
  - tar –xvf comGSI_v3.2.tar
  - tar –xvf CRTM_Coefficients-2.0.5.tar

System Requirements

- FORTRAN 90+ compiler
- C compiler
- Perl
- Gnu Make
- NetCDF V3.6+, & V4+
- Linear algebra library (ESSL or LAPACK/BLAS)
- MPI V1.2+ & OpenMP

Tour of the Directory Structure

Inside the top level of the comGSI_v3.2/ directory are four scripts and five directories.

- arch/
- clean
- compile
- configure
- fix/
- makefile
- run/
- src/
- util/

Build Infrastructure

- Uses DTC Build system
- **/arch** directory contains rules & scripts for build.
  - **/arch/Config.pl** perl script for parsing system info & combining together *configure.gsi* file.
  - **/arch/preamble**: uniform requirements for the code, such as word size, etc.
  - **/arch/configure.defaults** default platform settings
  - **/arch/postamble**: standard make rules & dependencies
- **./clean** script to clean the build.
- **./configure** script to create configuration file *configure.gsi*; contains info on compiler, MPI, & paths.
- **./compile** script to compile executable.
- **./makefile** top level makefile for build.

The rest

- **fix/** directory containing fixed parameter files
  - Background error covariance and observation errors
  - CRTM coefficients – moved to a separate directory due to size
  - Observation data control files
  - BUFR tables for Prepbufr files
- **run/**
  - run_gsi.ksh sample run script
  - gsi.exe executable
- **src/** source directory
  - **libs/** supplemental library source code
  - **main/** main GSI source code
- **util/** additional community tools

Supplemental Libraries (libs/)

- **bacio/** NCEP BACIO library
- **bufr/** NCEP BUFR library
- **crtm_2.0.5/** JCSDA Commuity Radiative Transfer Model
- **gfsio/** Unformatted Fortran record for GFS I/O
- **gsdcloud/** GSD Cloud Analysis
- **misc/** Misc additional libraries
- **nemsio/** Support for NEMS I/O
- **sfcio/** NCEP GFS surface file I/O module
- **sigio/** NCEP GFS atmospheric file I/O module
- **sp/** NCEP spectral-grid transforms (global application only)
- **w3/** NCEP W3 library (date/time manipulation, GRIB)

Building GSI

Building GSI

- Build sequence
  - ./clean –a
  - Set library paths
    - setenv WRF_DIR *Location_of_WRF_directory*
    - setenv LAPACK_PATH (*typically only needed for Linux w/ ifort or gfortran*).
  - ./configure
    - Customize file *configure.gsi* if necessary
  - ./compile
- Successful compilation will produce:
  - *comGSI_v3.2/run/gsi.exe*

Clean Compilation

- To remove all object files and executables, type:

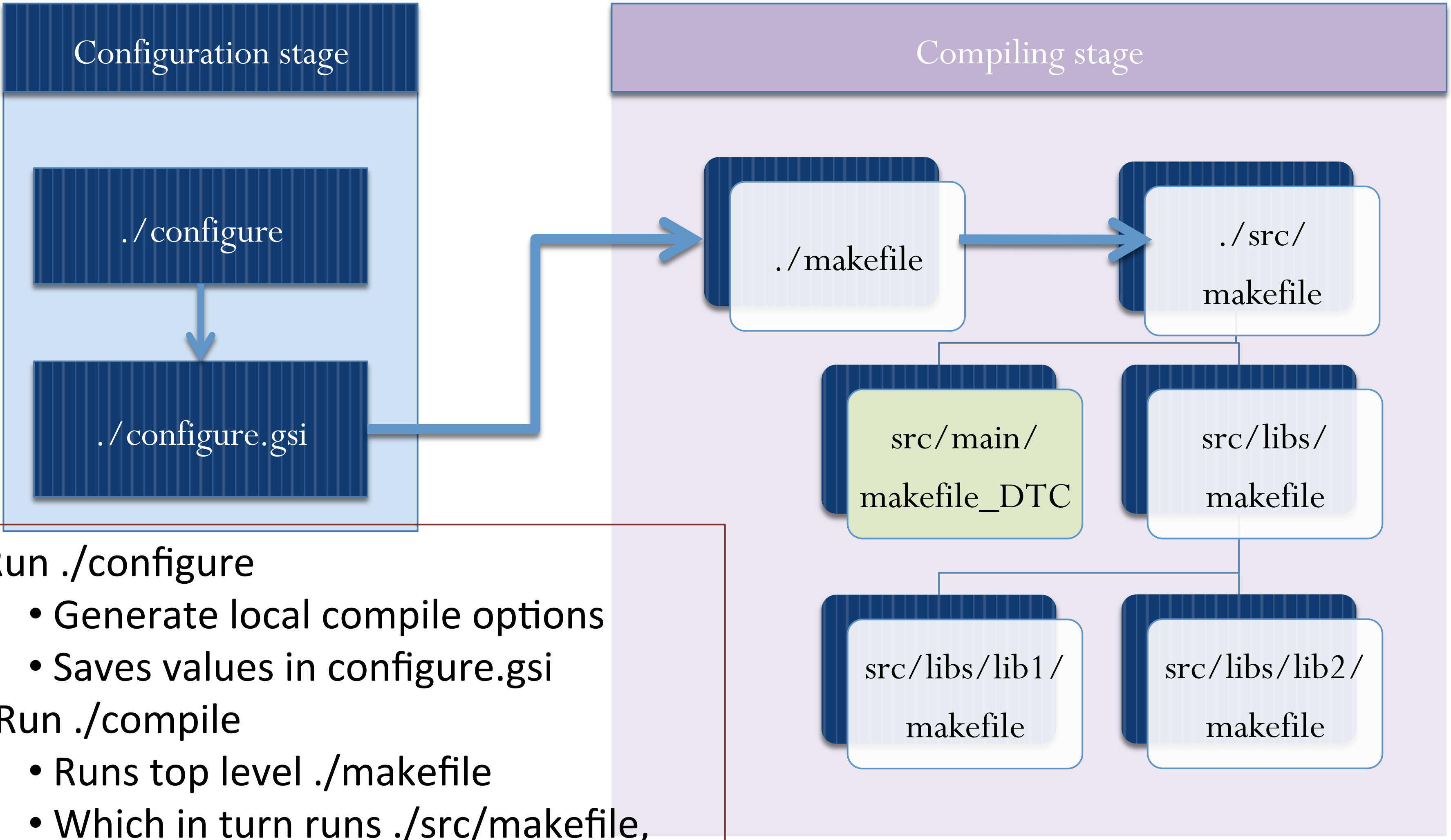
  clean

- To remove **all** built files, including the configure file, type: *clean –a*
  - A clean all needed if:
    - Compilation failed
    - Want to change configuration file

Diagnosing Build Issues

- How the build system works
- What to do when the build fails

GSI Build System



✓ Run ./configure

- Generate local compile options
- Saves values in configure.gsi

✓ Run ./compile

- Runs top level ./makefile
- Which in turn runs ./src/makefile,
- Which runs ./src/main/makefile

Developmental Testbed Center

How the build works

- Running *./configure* creates file configure.gsi by:
  - Running the Perl script /arch/Config.pl
  - Script Config.pl queries the system & selects the appropriate entry from /arch/configure.defaults
  - Results are saved to configure.gsi.

Identifying Build Errors

- Most build or run problems must be diagnosed by use of the log files.
- For build errors pipe the standard out and standard error into a log file with a command such as (for csh) ./compile |& tee build.log
- Search the log file for any instance of the word "Error." Its presence indicates a build error. Be certain to use the exact spelling with a capital "E."
- If the build fails, but the word "Error" is not present in the log file, it typically indicates a failure in link the phase. Information on the failed linking phase will be present at the very end of the log file.

Fixing Build Issues

- Most build problems are due to non-standard instillation of one of the following:
  - compiler,
  - mpi,
  - or support libraries.
- Edit paths in the file configure.gsi to correctly reflect your system.
- When the build succeeds, modify file arch/configure.defaults to include new settings.
- Please report issues to gsi_help so they can be addressed in next release.

Fixing Build Issues (continued)

- The name or location of your LAPACK library may differ from what the build assumes. See **MYLIBsys**
- You may also want to use different Fortran compiler flags: See **FFLAGS_***
- You may also want to use different C compiler flags: See **CFLAGS**
- You may have a slightly different name for your compilers: See **SFC**, **SF90**, and **SCC** to specify your Fortran, Fortran90+, and C compilers.
- See the User's Guide for details

configure.gsi

```
SHELL              =     /bin/sh
# Listing of options that are usually independent of machine type.
# When necessary, these are over-ridden by each architecture.
#### Architecture specific settings ####

# Settings for Linux x86_64, GNU compilers (gfortran & gcc)  (dmpar,optimize)#
LDFLAGS       =  -Wl,-noinhibit-exec

COREDIR       =  /d1/stark/GSI/src/intel/V3.2/release_V3.2
INC_DIR       =  $(COREDIR)/include
BYTE_ORDER     =  LITTLE_ENDIAN
SFC          =  gfortran
SF90          =  gfortran -ffree-form
SCC          =  gcc
INC_FLAGS      =  -I $(INC_DIR) -I /usr/local/netcdf3-ifort/include
```

configure.gsi

```
SHELL              =     /bin/sh
# Listing of options that are usually independent of machine type.
# When necessary, these are over-ridden by each architecture.
#### Architecture specific settings ####

# Settings for Linux x86_64, GNU compilers (gfortran & gcc)  (dmpar,optimize)#
LDFLAGS       =  -Wl,-noinhibit-exec

COREDIR       =  /d1/stark/GSI/src/intel/V3.2/release_V3.2
INC_DIR       =  $(COREDIR)/include
BYTE_ORDER     =  LITTLE_ENDIAN
SFC          =  gfortran
SF90          =  gfortran -ffree-form
SCC          =  gcc
INC_FLAGS      =  -I $(INC_DIR) -I /usr/local/netcdf3-ifort/include
```

configure.gsi

```
SHELL              =     /bin/sh
# Listing of options that are usually independent of machine type.
# When necessary, these are over-ridden by each architecture.
#### Architecture specific settings ####

# Settings for Linux x86_64, GNU compilers (gfortran & gcc)  (dmpar,optimize)#
LDFLAGS       =  -Wl,-noinhibit-exec

COREDIR       =  /d1/stark/GSI/src/intel/V3.2/release_V3.2
INC_DIR       =  $(COREDIR)/include
BYTE_ORDER     =  LITTLE_ENDIAN
SFC          =  gfortran
SF90          =  gfortran -ffree-form
SCC          =  gcc
INC_FLAGS      =  -I $(INC_DIR) -I /usr/local/netcdf3-ifort/include
```

configure.gsi

```
SHELL            =    /bin/sh
# Listing of options that are usually independent of machine type.
# When necessary, these are over-ridden by each architecture.
#### Architecture specific settings ####

# Settings for Linux x86_64, GNU compilers (gfortran & gcc)  (dmpar,optimize)#
LDFLAGS      =  -Wl,-noinhibit-exec

COREDIR      =  /d1/stark/GSI/src/intel/V3.2/release_V3.2
INC_DIR      =  $(COREDIR)/include
BYTE_ORDER     =  LITTLE_ENDIAN
SFC          =  gfortran
SF90          =  gfortran -ffree-form
SCC          =  gcc
INC_FLAGS      =  -I $(INC_DIR) -I /usr/local/netcdf3-ifort/include
```

Developmental Testbed Center

configure.gsi

```
SHELL              =     /bin/sh
# Listing of options that are usually independent of machine type.
# When necessary, these are over-ridden by each architecture.
#### Architecture specific settings ####

# Settings for Linux x86_64, GNU compilers (gfortran & gcc)  (dmpar,optimize)#
LDFLAGS       =  -Wl,-noinhibit-exec

COREDIR       =  /d1/stark/GSI/src/intel/V3.2/release_V3.2
INC_DIR       =  $(COREDIR)/include
BYTE_ORDER    =  LITTLE_ENDIAN
SFC           =  gfortran
SF90          =  gfortran -ffree-form
SCC           =  gcc
INC_FLAGS     =  -I $(INC_DIR) -I /usr/local/netcdf3-ifort/include
```

configure.gsi

Fortran Build Flags:

```
FFLAGS_i4r4    =
FFLAGS_i4r8    =  -fdefault-real-8
FFLAGS_i8r8    =  -fdefault-integer-8 -fdefault-real-8
FFLAGS_DEFAULT =  -fno-second-underscore -fno-range-check
#FFLAGS_DEBUG   =  -g -O0 -C
FFLAGS_FULLOPT =  -O3
FFLAGS_OPT     =  $(FFLAGS_FULLOPT) $(FFLAGS_DEBUG)
FFLAGS         =  $(FFLAGS_OPT) $(FFLAGS_DEFAULT) $(INC_FLAGS) $(LDFLAGS)
```

configure.gsi

```
# Library build flags
FFLAGS_BACIO   =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT)
FFLAGS_BUFR    =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT) $(FFLAGS_i4r8)
CFLAGS_BUFR    =  $(FFLAGS_FULLOPT) -DUNDERSCORE
FFLAGS_CLOUD   =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT)
FFLAGS_CRTM    =  $(FFLAGS_DEFAULT)
FFLAGS_GFSIO   =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT) $(FFLAGS_i4r4)
FFLAGS_SFCIO   =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT) $(FFLAGS_i4r4)
FFLAGS_SIGIO   =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT) $(FFLAGS_i4r4)
FFLAGS_SP      =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT) $(FFLAGS_i4r8)
FFLAGS_W3      =  $(FFLAGS_FULLOPT) $(FFLAGS_DEFAULT)
```

configure.gsi

```
CPP            =  cpp
CPP_FLAGS      =  -C -P -D$(BYTE_ORDER) -D_REAL8_ -DWRF -DLINUX -DPGI
CPP_F90FLAGS   =  -traditional-cpp -lang-fortran

# MPI compiler wrappers
DM_FC          =  mpif90
DM_F90         =  mpif90 -ffree-form
DM_CC          =  gcc

CFLAGS         =  -O0 -DLINUX -DUNDERSCORE
CFLAGS2        =  -DLINUX -Dfunder -DFortranByte=char -DFortranInt=int -DFortranLlong='long long'
```

configure.gsi

```
#       Macros, these should be generic for all machines
LN              =  ln -sf
MAKE                =  make -i -r
RM               =  /bin/rm -f
CP              =  /bin/cp
AR              =  ar
MKDIR               =  /bin/mkdir –p
```

On platforms such as the IBM it is sometimes necessary to modify these paths to point to the gnu version of the Unix tools rather than the XLF version.

configure.gsi

```
MYLIBsys       =  -llapack -lblas
NETCDF_PATH    =  /usr/local/netcdf3-ifort/lib
```

The main library path of interest is the one to the LAPACK and BLAS libraries. Common issues are that:

- ***library names are incorrect***
- ***library paths are incorrect***
- ***or both***

Check that your system has libraries in the specified path and with the specified names.

configure.gsi

Library Paths

```
NETCDFPATH     =  /usr/local/netcdf3-ifort
NETCDFLIBS     =  -L$(NETCDFPATH) -lnetcdff -lnetcdf
WRF_DIR        =  /d1/stark/WRF/intel/WRFV3.3.1_arw
```

Check that your system has libraries in the specified path and with the specified names.

Getting Help

- For more detailed information on installation see: GSI User's Guide, chapter 2
  - www.dtcenter.org/com-GSI/users/docs/index.php
- Check the FAQ
  - www.dtcenter.org/com-GSI/users/support/faqs/index.php
- Check the Known Issues
  - www.dtcenter.org/com-GSI/users/support/known_issues/index_v3.2.php
- For further assistance contact:
  - gsi_help@ucar.edu