

# The NetCDF Installation and Porting Guide

---

NetCDF Version 3.6.0  
December 2004

Russ Rew, John Caron, and Ed Hartnett  
Unidata Program Center

---

Copyright © 2004 University Corporation for Atmospheric Research

Permission is granted to make and distribute verbatim copies of this manual provided that the copyright notice and these paragraphs are preserved on all copies. The software and any accompanying written materials are provided “as is” without warranty of any kind. UCAR expressly disclaims all warranties of any kind, either expressed or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The Unidata Program Center is managed by the University Corporation for Atmospheric Research and sponsored by the National Science Foundation. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Mention of any commercial company or product in this document does not constitute an endorsement by the Unidata Program Center. Unidata does not authorize any use of information from this publication for advertising or publicity purposes.

# Table of Contents

<b>1</b>	<b>Quick Instructions for Installing NetCDF on Unix</b>	<b>1</b>
<b>2</b>	<b>Building and Installing NetCDF on Unix Systems</b>	<b>3</b>
2.1	Installation Requirements	3
2.2	Specifying the Environment for Building	4
2.3	Building on 64 Bit Platforms	5
2.4	Running the configure Script	5
2.5	Running make	6
2.6	Testing the Build	6
2.7	Installing NetCDF	7
2.8	Platform Specific Notes	7
2.8.1	AIX	7
2.8.2	Cygwin	7
2.8.3	HPUX	7
2.8.4	Irix	8
2.8.5	Linux	8
2.8.6	Macintosh	8
2.8.7	OSF1	8
2.8.8	SunOS	9
2.9	Additional Porting Notes	9
<b>3</b>	<b>Building and Installing NetCDF on Windows</b>	<b>11</b>
3.1	Getting Prebuilt netcdf.dll	11
3.2	Installing the DLL	11
3.3	Building netcdf.dll with VC++ 6.0	12
3.4	Using netcdf.dll with VC++ 6.0	13
3.5	Building netcdf.dll with VC++.NET	14
3.6	Using netcdf.dll with VC++.NET	14
<b>4</b>	<b>If Something Goes Wrong</b>	<b>15</b>
4.1	Troubleshooting	15
4.2	Finding Help On-line	16
4.3	Reporting Problems	16
	<b>Index</b>	<b>17</b>



# 1 Quick Instructions for Installing NetCDF on Unix

Who has time to read long installation manuals these days?

To install netCDF, uncompress and unpack the tar file, then change to the src directory:

```
gunzip netcdf-3_6_0.tar.gz
tar -xf netcdf-3_6_0.tar
cd netcdf-3_6_0/src
```

Now run the usual configure, make test, make install cycle:

```
./configure
make test
make install
```

The configure script will try to find necessary tools in your path. When you run configure you may optionally use the `--prefix` argument to change the default installation directory. For, the following will install the library in `/usr/local/lib`, the header file in `/usr/local/include`, and the utilities in `/usr/local/bin`.

```
./configure --prefix=/usr/local
```

The default install root is `..` (i.e. the parent directory, which will be `netcdf-3.6.0`).

If all this doesn't work, then you might have to read the next chapter. Better luck next time!



## 2 Building and Installing NetCDF on Unix Systems

The latest version of this document is available at [http://www.unidata.ucar.edu/packages/netcdf/INSTALL\\_beta/index.html](http://www.unidata.ucar.edu/packages/netcdf/INSTALL_beta/index.html).

This document contains instructions for building and installing the netCDF package from source on various platforms. Prebuilt binary releases are (or soon will be) available for various platforms from <http://www.unidata.ucar.edu/packages/netcdf/binaries.html>.

This document describes installation of version 3.6.0 (beta) of the netCDF library, for information about installing earlier versions of netCDF, see <http://www.unidata.ucar.edu/packages/netcdf/INSTALL.html>.

### 2.1 Installation Requirements

Depending on the platform, you may need up to 25 Mbytes of free space to unpack, build, and run the tests. You will also need a Standard C compiler. If you have compilers for FORTRAN 77, FORTRAN 90, or C++, the corresponding netCDF language interfaces may also be built and tested. Compilers and associated tools will only be found if they are in your path.

If you want to run the large file tests, you will need about 13 GB of free disk space, as some very large files are created. The created files are immediately deleted after the tests complete. These large file tests are not run as part of the make test step; they are only run for make extra\_test.

If you wish to build from source on a Windows (Win32) platform, different instructions apply. See [Chapter 3 \[Building and Installing NetCDF on Windows\], page 11](#).

To fully work with the netCDF source code, several extra utilities are required to fully build everything from source. If you are going to modify the netCDF source code, you will need some or all of the following tools. All are freeware.

- m4** Macro processing language used heavily in libsrc, nc\_test. Generates (in these cases) C code from m4 source. Version 1.4 works fine with release 3.5.1 and 3.6.0.
- nm** Lists contents of an “object” file. GNU nm does not mix well with vendor compilers in the 64-bit world, so make sure that you are using GNU nm with GNU compilers, or a vendor nm with your vendor compiler.
- ar** Creates libraries. GNU ar does not mix well with vendor compilers in the 64-bit world, so make sure that you are using GNU ar with GNU compilers, or a vendor ar with your vendor compiler.

The following tools are not required to build netCDF. They may be needed if you intend to work with the netCDF source code as a developer.

#### **flex and yacc**

Used in ncgen directory to parse CDL files. Generates C files from .y and .l files. You only need to use this to modify ncgen’s understanding of CDL grammar.

- makeinfo** Generates all documentation formats (except man pages) from texinfo source. I’m using makeinfo version 4.2, as of release 3.6.0. If you have trouble with

makeinfo, upgrade to at least 4.2 and try again. You only need makeinfo if you want to modify the documentation.

**autoconf** Generates the configure script. Autoconf is only needed to modify the configure script. Version 2.59 or later is required. Automake is not used with netCDF version 3.6.

The most recent version of all netCDF documents can always be found at the netCDF website. <http://www.unidata.ucar.edu/packages/netcdf/>.

## 2.2 Specifying the Environment for Building

The netCDF configure script will set some environment variables that are important for building from source code. It is only necessary to set them to override default behavior.

The netCDF configure script searches your path to find the compilers and tools it needed. To use compilers that can't be found in your path, set their environment variables.

When finding compilers, vendor compilers will be preferred to GNU compilers. Not because we don't like GNU, but because we assume if you purchased a compiler, you want to use it. Setting `CC` allows you to over-ride this preference. (Alternatively, you could temporarily remove the compiler's directories from your `PATH`.)

For example, on an AIX system, configure will first search for `xlC`, the AIX compiler. If not found, it will try `gcc`, the GNU compiler. To override this behavior, set `CC` to `gcc` (in sh: `export CC=gcc`). (But don't forget to also set `CXX` to `g++`, or else configure will try and use `xlC`, the AIX C++ compiler.)

By default, the netCDF library is built with assertions turned on. If you wish to turn off assertions, set `CPPFLAGS` to `-DNDEBUG` (csh ex: `setenv CPPFLAGS -DNDEBUG`).

### Variable Description Notes

<code>CC</code>	C compiler		If you don't specify this, the configure script will try to find a suitable C compiler such as <code>cc</code> , <code>c89</code> , <code>xlC</code> , or <code>gcc</code> .
<code>FC</code>	Fortran compiler (if any)		If you don't specify this, the configure script will try to find a suitable Fortran 90 or Fortran 77 compiler. Set <code>FC</code> to "" explicitly, if no Fortran interface is desired.
<code>F90</code>	Fortran compiler (if any)	90	If you don't specify this, the configure script will try to find a suitable Fortran 90 compiler. Not needed if <code>FC</code> specifies a Fortran 90 compiler. Set <code>F90</code> to "" explicitly, if no Fortran 90 interface desired. For a vendor F90 compiler, make sure you're using the same vendor's F77 compiler. Using Fortran compilers from different vendors, or mixing vendor compilers with <code>g77</code> , the GNU F77 compiler, is not supported and may not work.

CXX	C++ compiler	If you don't specify this, the configure script will try to find a suitable C++ compiler. Set CXX to "" explicitly, if no C++ interface is desired. If using a vendor C++ compiler, use that vendor's C compiler to compile the C interface. Using different vendor compilers for C and C++ may not work.
CFLAGS	C compiler flags	"-O" or "-g", for example.
CPPFLAGS	C preprocessor options	"-DNDEBUG" to omit assertion checks, for example.
FFLAGS	Fortran compiler flags	"-O" or "-g", for example.
F90FLAGS	Fortran 90 compiler flags	"-O" or "-g", for example. If you don't specify this, the value of FFLAGS will be used.
CXXFLAGS	C++ compiler flags	"-O" or "-g", for example.
ARFLAGS, NMFLAGS, FPP, M4FLAGS, LIBS, FLIBS, FLDFLAGS	Miscellaneous	One or more of these were needed for some platforms, as specified below. Unless specified, you should not set these environment variables, because that may interfere with the configure script.

The section marked Tested Systems below contains a list of systems on which we have built this package, the environment variable settings we used, and additional commentary.

## 2.3 Building on 64 Bit Platforms

Some platforms support special options to build in 64-bit mode.

NetCDF 3.6.0 beta has been tested as 64-bit builds on SunOS and AIX. The options needed to build in 64-bit mode are described here.

AIX	Set -q64 option in all compilers, and set NMFLAGS to -X32, and ARFLAGS to '-X32 cru'. Alternatively, set environment variable OBJECT_MODE to 64 before running configure.
IRIX	Set the -64 option in all compilers.
SunOS	Use the -xarch=v9 flag on all compilers. This is not supported on the x86 platform.

## 2.4 Running the configure Script

To create the Makefiles needed to build netCDF, you must run the provided configure script. Go to the top-level netCDF src/ directory.

Decide where you want to install this package. Use this for the "--prefix=" argument to the configure script below. The default installation prefix is "..", which will install the package's files in ../bin, ../lib, and ../man relative to the netCDF src/ directory.

Execute the configure script:

```
./configure --prefix=whatever_you_decided
```

The "--prefix=..." specification is optional; if omitted, "." designating the parent directory will be used as a default.

The configure script will examine your computer system – checking for attributes that are relevant to building the netCDF package. It will print to standard output the checks that it makes and the results that it finds.

The configure script will also create the file "config.log", which will contain error messages from the utilities that the configure script uses in examining the attributes of your system. Because such an examination can result in errors, it is expected that "config.log" will contain error messages. Therefore, such messages do not necessarily indicate a problem (a better indicator would be failure of the subsequent "make"). One exception, however, is an error message in "config.log" that indicates that a compiler could not be started. This indicates a severe problem in your compilation environment – one that you must fix.

## 2.5 Running make

Run "make". This will build one or more netCDF libraries. It will build the basic netCDF library libsrc/libnetcdf.a. If you have Fortran 77 or Fortran 90 compilers, then the Fortran interfaces will be included in this library. If you have a C++ compiler, then the C++ interface will be built into the library cxx/libnetcdf.c++.a. This will also build the netCDF utilities ncgen(1) and ncdump(1).

Run make like this:

```
make
```

## 2.6 Testing the Build

Run "make test" to verify that the netCDF library and executables have been built properly. This will build and run various test programs that test the C, Fortran, and C++ interfaces as well as the "ncdump" and "ncgen" utility programs. Lines in the output beginning with "\*\*\*\*" report on success or failure of the tests; any failures will be reported before halting the test. Compiler and linker warnings during the testing may be ignored.

Run the tests like this:

```
make test
```

If you plan to use the 64-bit offset format (introduced in version 3.6.0) to create very large files (i.e. larger than 2 GB), you should probably run "make extra\_test" which tests the large file features. You must have 13 GB of free disk space for these tests to successfully run. (The test files are deleted when the test completes, so you get your disk space back.) You may wish to set environment variable TEMP\_LARGE to a directory to which large files can be written. (For example, in csh: setenv TEMP\_LARGE /ptmp/edh).

Run the large file tests like this:

```
make extra_test
```

If the tests fail See [Chapter 4 \[If Something Goes Wrong\]](#), page 15.

## 2.7 Installing NetCDF

To install the libraries and executables, run "make install". This will install to the directory specified in the configure step, or to ../lib (that is, it will create a lib directory under the netcdf-3.6.0 directory, and install the library there.)

Run the installation like this:

```
make install
```

Try linking your applications. Let us know if you have problems (see [Section 4.3 \[Reporting Problems\]](#), page 16). Port the library to other platforms. Share data.

## 2.8 Platform Specific Notes

The following platform-specific note may be helpful when building and installing netCDF. Consult your vendor manuals for information about the options listed here. Compilers can change from version to version; the following information may not apply to your platform.

Full output from some of the platforms of the test platforms for netCDF 3.6.0 can be found at <http://www.unidata.ucar.edu/packages/netcdf/builds/>.

### 2.8.1 AIX

We found the vendor compilers in /usr/vac/bin, and included this in our PATH. Compilers were xlc, xlf, xlf90, xlc.

The F90 compiler requires the qsuffix option to believe that F90 code files can end with .f90. This is automatically turned on by configure when needed (we hope):

```
F90FLAGS=-qsuffix=f=f90
```

To compile 64-bit code, set the environment variable OBJECT\_MODE to 64, or use the -q64 option on all AIX compilers by setting CFLAGS, FFLAGS, and CXXFLAGS to -q64.

The following is also necessary on an IBM AIX SP system for 64-bit mode:

```
ARFLAGS='-X64 cru'
NMFLAGS='-X64'
```

There are thread-safe versions of the AIX compilers. For example, xlc\_r is the thread-safe C compiler. The NetCDF configure script ignores these compilers. To use thread-safe compilers, override the configure script by setting CC to xlc\_r; similarly for FC and CXX.

For large file support, AIX requires that the macro \_LARGE\_FILES be defined. The configure script does this using AC\_SYS\_LARGEFILES. Unfortunately, this misfires when OBJECT\_MODE is 64, or the q64 option is used. The netCDF tries to fix this by turning on \_LARGE\_FILES anyway in these cases.

The GNU C compiler does not mix successfully with the AIX fortran compilers.

### 2.8.2 Cygwin

NetCDF builds under Cygwin tools on Windows just as with Linux.

### 2.8.3 HPUX

The HP Fortran compiler (f77, a.k.a. fort77) requires FLIBS to include -IU77 for the fortran tests to work. The configure script does this automatically.

For the c89 compiler to work, CPPFLAGS must include `-D_HPUX_SOURCE`. This isn't required for the cc compiler. The configure script adds this as necessary.

For large file support, HP-UX requires `_FILE_OFFSET_BITS=64`. The configure script sets this automatically.

The HP-UX C++ compiler doesn't work on netCDF code. It's too old for that. So either use GNU to compile netCDF, or skip the C++ code by setting CXX to "" (in csh: `setenv CXX ""`).

Building a 64 bit version may be possible with the following settings:

```
CC=/bin/cc
CPPFLAGS='-D_HPUX_SOURCE -D_FILE_OFFSET_BITS=64'    # large file support
CFLAGS='-g +DD64'                                  # 64-bit mode
FC=/opt/fortran90/bin/f90                          # Fortran-90 compiler
FFLAGS='-w +noppu +DA2.0W'                         # 64-bit mode, no "_" suffixes
FLIBS=-lU77
CXX=''
```

## 2.8.4 Irix

A 64-bit version can be built by setting CFLAGS, FFLAGS, and CXXFLAGS to `-64`.

On our machine, there is a `/bin/cc` and a `/usr/bin/cc`, and the `-64` option only works with the former.

## 2.8.5 Linux

The `f2cFortran` flag is required with GNU fortran:

```
CPPFLAGS=-Df2cFortran
```

For Portland Group Fortran, set `pgiFortran` instead:

```
CPPFLAGS=-DpgiFortran
```

Portland Group F90/F95 does not mix with GNU g77.

The netCDF configure script should notice which fortran compiler is being used, and set these automatically.

For large file support, `_FILE_OFFSET_BITS` must be set to 64. The netCDF configure script should set this automatically.

## 2.8.6 Macintosh

The `f2cFortran` flag is required with GNU fortran (`CPPFLAGS=-Df2cFortran`). The NetCDF configure script should and set this automatically.

For IBM compilers on the Mac, the following may work (we lack this test environment):

```
CC=/usr/bin/cc
CPPFLAGS=-DIBM2Fortran
FC=xlf
F90=xlf90
F90FLAGS=-qsuffix=cpp=f90
```

## 2.8.7 OSF1

NetCDF builds out of the box on OSF1.

### 2.8.8 SunOS

PATH should contain /usr/ccs/bin to find make, nm, ar, etc.

For large file support, `_FILE_OFFSET_BITS` must be 64. Configure will turn this on automatically.

Large file support doesn't work with c89, unless the `-Xa` option is used. The netCDF configure script turns this on automatically where appropriate.

To compile in 64-bit mode, use option `-xarch=v9` on all compilers (i.e. in `CFLAGS`, `FFLAGS`, and `CXXFLAGS`).

When compiling with GNU Fortran (g77), the `-Df2cFortran` flag is required for the Fortran interface to work. The NetCDF configure script turns this on automatically if needed.

## 2.9 Additional Porting Notes

The configure and build system should work on any system which has a modern "sh" shell, "make", and so on. The configure and build system is less portable than the "C" code itself, however. You may run into problems with the "include" syntax in the Makefiles. You can use GNU make to overcome this, or simply manually include the specified files after running configure.

If you can't run the configure script, you will need to create `libsrc/nconfig.h` and `fortran/nfconfig.inc`. Start with `libsrc/nconfig.in` and `fortran/nfconfig.in` and set the defines as appropriate for your system.

Operating system dependency is isolated in the "ncio" module. We provide two versions. `posixio.c` uses POSIX system calls like `"open()"`, `"read()"` and `"write()"`. `ffio.c` uses a special library available on CRAY systems. You could create other versions for different operating systems. The program `"t_ncio.c"` can be used as a simple test of this layer.

Numerical representation dependency is isolated in the "ncx" module. As supplied, `ncx.m4` (`ncx.c`) supports IEEE floating point representation, VAX floating point, and CRAY floating point. `BIG_ENDIAN` vs `LITTLE_ENDIAN` is handled, as well as various sizes of "int", "short", and "long". We assume, however, that a "char" is eight bits.

There is a separate implementation of the ncx interface available as `ncx_cray.c` which contains optimizations for CRAY vector architectures. Move the generic `ncx.c` out of the way and rename `ncx_cray.c` to `ncx.c` to use this module. By default, this module does not use the `IEG2CRAY` and `CRAY2IEG` library calls. When compiled with aggressive in-lining and optimization, it provides equivalent functionality with comparable speed and clearer error semantics. If you wish to use the IEG library functions, compile this module with `-DUSE_IEG`.



## 3 Building and Installing NetCDF on Windows

NetCDF can be built and used from a variety of development environments on Windows. The netCDF library is implemented as a Windows dynamic link library (DLL). The simplest way to get started with netCDF under Windows is to download the pre-built DLL from the Unidata web site.

Instructions are also given for building the netCDF DLL from the source code.

VC++ documentation being so voluminous, finding the right information can be a chore. There's a good discussion of using DLLs called "About Dynamic-Link Libraries" at (perhaps) [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/dynamic\\_link\\_libraries.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/dynamic_link_libraries.asp).

From the .NET point of view, the netCDF dll is unmanaged code. As a starting point, see the help topic "Consuming Unmanaged DLL Functions" which may be found at <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconConsumin>.

### 3.1 Getting Prebuilt netcdf.dll

We have pre-built Win32 binary versions of the netcdf dll and static library, as well as ngen.exe and ncdump.exe (dll and static versions). You can get them from <ftp://ftp.unidata.ucar.edu/pub/netcdf/contrib/win32/netcdf-3.5.0.win32bin.zip>. (Note: we don't have a C++ interface here).

### 3.2 Installing the DLL

Whether you get the pre-built DLL or build your own, you'll then have to install it somewhere so that your other programs can find it and use it.

There are standard places to put DLLs, the Windows\System folder (Windows 98/ME) or Windows\System32 folder (Windows2000/XP), or you can leave them wherever you want, and every development project that uses the dll will have to be told to search the netCDF directory when it's linking.

On the .NET platform, there is also the global assembly cache (see MSDN topic "Global Assembly Cache").

Following Windows conventions, the netCDF files belong in the following places:

File(s)	Description	Location
netcdf.dll	C and Fortran function in DLL	Windows\System (98/ME) or Windows\System32 (2000/XP)
netcdf.lib	Library file?	Windows\System (98/ME) or Windows\System32 (2000/XP)

ncgen.exe, ncdump.exe	NetCDF utilities	Windows\System (98/ME) or Windows\System32 (2000/XP)
netcdf-3\src	netCDF source code	Program Files\Unidata

### 3.3 Building netcdf.dll with VC++ 6.0

To build the library yourself, get the file <ftp://ftp.unidata.ucar.edu/pub/netcdf/contrib/win32/netcdf-3.5.0.win32make.VC6.zip>

The makefiles there describe how to build netcdf-3.5 using the using Microsoft Visual C++ 6.x and (optionally) Digital Visual Fortran 6.x. Because of difficulties in getting Microsoft Visual Studio to fall in line with our existing source directory scheme, we chose `_not_` to build the system "inside" Visual Studio. Instead, we provide a simple group of "msoft.mak" files which can be used. If you wish to work in Visual Studio, go ahead. Read the section called "Macros" at the end of this discussion.

As of this writing, we have not tried compiling the C++ interface in this environment.

`nmake` is a Microsoft version of `make`, which comes with VC 6.0 (and VC 7.0) in directory `C:\Program Files\Microsoft Visual Studio\VC98\Bin` (or, for VC 7.0, `C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\bin`).

To build netcdf, proceed as follows:

```
unpack source distribution.
```

```
copy netcdf-3.5.0.win32make.VC6.zip
```

copy netcdf-3.5.0.win32make.VC6.zip into the netcdf-3.5.0/src directory, and unzip it from there.

```
cd src\libsrc; nmake /f msoft.mak
```

Run this command in `src\libsrc`. This will build `netcdf.lib` and `netcdf.dll` Note: This makefiles make DLLs. To make static libraries see section on static libraries.

```
nmake /f msoft.mak test
```

Optionally, in `src\libsrc`, make and run the simple test.

```
cd ..\fortran; nmake /f msoft.mak
```

Optionally build the fortran interface and rebuild dll in `..\libsrc` to include the fortran interface. Note Bene: We don't provide a `.DEF` file, so this step changes the "ordinals" by which entry points in the DLL found. Some sites may wish to modify the `msoft.mak` file(s) to produce a separate library for the fortran interface.

```
nmake /f msoft.mak test
```

(necessary if you want to use fortran code) While you are in `src\fortran`; `nmake /f msoft.mak test` This tests the netcdf-2 fortran interface.

```
cd ..\nctest; nmake /f msoft.mak test
    (optional, but recommended) In src\nctest; nmake /f msoft.mak test This tests
    the netcdf-2 C interface.
```

```
cd ..\nc_test; nmake /f msoft.mak test
    (optional, but highly recommended) In src\nc_test; nmake /f msoft.mak test
    This tortures the netcdf-3 C interface.
```

```
cd ..\nf_test; nmake /f msoft.mak test
    (optional, but highly recommended if you built the fortran interface) In
    src\nf_test; nmake /f msoft.mak test This tortures the netcdf-3 fortran
    interface.
```

```
..\ncdump; nmake /f msoft.mak
    In src\ncdump; nmake /f msoft.mak This makes ncdump.exe.
```

```
..\ncgen; nmake /f msoft.mak
    In src\ncgen; nmake /f msoft.mak This makes ncgen.exe.
```

```
..\ncdump; nmake /f msoft.mak test
    (optional) In src\ncdump; nmake /f msoft.mak test This tests ncdump. Both
    ncgen and ncdump need to be built prior to this test. Note the makefile sets
    the path so that ..\libsrc\netcd.dll can be located.
```

```
..\ncgen; nmake /f msoft.mak test
    (optional) In src\ncgen; nmake /f msoft.mak test This tests ncgen. Both ncgen
    and ncdump need to be built prior to this test. Note the makefile sets the path
    so that ..\libsrc\netcd.dll can be located.
```

**To Install**

Copy libsrc\netcd.lib to a LIBRARY directory. Copy libsrc\netcd.h and fortran/netcd.inc to an INCLUDE directory. Copy libsrc\netcd.dll, ncdump/ncdump.exe, and ncgen/ncgen.exe to a BIN directory (someplace in your PATH).

**3.4 Using netcd.dll with VC++ 6.0**

To use the netcd.dll:

1. Place these in your include directory: netcd.h C include file netcd.inc Fortran include file

- 2a. To use the Dynamic Library (shared) version of the netcd library: Place these in a directory that's in your PATH: netcd.dll library dll ncgen.exe uses the dll ncdump.exe uses the dll

Place this in a library directory to link against: netcd.lib library

- 2b. Alternatively, to use a static version of the library

Place this in a library directory to link against: netcds.lib library

Place these in a directory that's in your PATH: ncgens.exe statically linked (no DLL needed) ncdumps.exe statically linked (no DLL needed)

### 3.5 Building netcdf.dll with VC++.NET

To build the netCDF dll with VC++.NET open the win32/NET/netcdf.sln file with Visual Studio. Both Debug and Release configurations are available - select one and build.

The resulting netcdf.dll file will be in subdirectory Release or Debug.

The netCDF tests will be built and run as part of the build process. The Fortran 77 interface will be built, but not the Fortran 90 or C++ interfaces.

### 3.6 Using netcdf.dll with VC++.NET

Load-time linking to the DLL is the most straightforward from C++. This means the netcdf.lib file has to end up on the compile command line. This being Windows, that's hidden by a GUI.

In Visual Studio 2003 this can be done by modifying three of the project's properties.

Open the project properties window from the project menu. Go to the linker folder and look at the general properties. Modify the property "Additional Library Directories" by adding the directory which contains the netcdf.dll and netcdf.lib files. Now go to the linker's input properties and set the property "Additional Dependencies" to netcdf.lib.

Finally, still within the project properties window, go to the C/C++ folder, and look at the general properties. Modify "Additional Include Directories" to add the directory with the netcdf.h file.

Now use the netCDF functions in your C++ code. Of course any C or C++ file that wants to use the functions will need:

```
#include <netcdf.h>
```

## 4 If Something Goes Wrong

The netCDF package is designed to build and install on a wide variety of platforms, but doesn't always. When the automatic install doesn't work, first see if the problem is something obvious (see [Section 4.1 \[Troubleshooting\]](#), page 15). If that doesn't help, try seeing if your problem has already been solved by someone else (see [Section 4.2 \[Finding Help\]](#), page 16). If that doesn't help, report your problem to Unidata, but please make sure you submit all the information we need to help (see [Section 4.3 \[Reporting Problems\]](#), page 16).

### 4.1 Troubleshooting

If the `./configure;make` test fails, it's a good idea to turn off the C++ and Fortran interfaces, and try to build the C interface alone. All other interfaces depend on the C interface, so nothing else will work until the C interface works. To turn off C++ and Fortran, set environment variables `CXX` and `FC` to `NULL` before running the netCDF configure script (with `csh: setenv FC "";setenv CXX ""`).

If the netCDF configure fails, most likely the problem is with your development environment. The configure script looks through your path to find all the tools it needs to build netCDF, including C compiler and linker, the `ar`, `ranlib`, and others. The configure script will tell you what tools it's found, and where they are on your system. Here's part of configure's output on a Linux machine:

```
checking CPPFLAGS... -Df2cFortran
checking CC CFLAGS... cc -g
checking which cc... /usr/bin/cc
checking CXX... c++
checking CXXFLAGS... -g -O2
checking which c++... /usr/local/bin/c++
checking FC... f77
checking FFLAGS...
checking which f77... /usr/bin/f77
checking F90... unset
checking AR... ar
checking ARFLAGS... cru
checking which ar... /usr/bin/ar
checking NM... nm
checking NMFLAGS...
checking which nm... /usr/bin/nm
checking RANLIB... ranlib
checking RANLIBFLAGS...
checking which ranlib... /usr/bin/ranlib
```

Make sure that the tools, directories, and flags are set to reasonable values, and compatible tools. For example the GNU tools may not inter-operate well with vendor tools. If you're using a vendor compiler, use the `ar`, `nm`, and `ranlib` that the vendor supplied.

As configure runs, it creates a `config.log` file. If configure crashes, do a text search of `config.log` for thing it was checking before crashing. If you have a licensing or tool compatibility problem, it will be obvious in `config.log`.

If the configure script runs, but the compile step doesn't work, or the tests don't complete successfully, the problem is probably in your CFLAGS or CPPFLAGS.

If you are planning on using large files (i.e. > 2 GiB), then make sure you run `make extra_test` to ensure that large files work on your system.

## 4.2 Finding Help On-line

The replies to all netCDF support emails are on-line and can be searched. Before reporting a problem to Unidata, please search this on-line database to see if your problem has already been addressed in a support email.

To search the support database, see [netcdf-support-search-url](#).

The netCDF mailing list also can be searched; see [netcdf-list-search-url](#).

## 4.3 Reporting Problems

To help us solve your problem, please include the following information in your email to [support@unidata.ucar.edu](mailto:support@unidata.ucar.edu).

Unfortunately, we can't solve build questions without this information; if you ask for help without providing it, we're just going to have to ask for it.

So why not send it immediately, and save us both the extra trouble?

1. the exact version of netCDF - see the `src/VERSION` file.
2. the \*complete\* output of `./configure`, `make`, and `make test`. Yes, it's long, but it's all important.
3. if the configure failed, the contents of `config.log`.
4. if you are having problems with very large files (larger than 2GiB), the output of `make extra_test`.

# Index

## 6

64-bit platforms ..... 5

## A

AIX 64-bit build ..... 5

ar ..... 3

autoconf ..... 4

## B

big endian ..... 9

binaries, windows ..... 11

binary releases ..... 3

bugs, reporting ..... 16

## C

config.log ..... 5

configure, running ..... 5

CRAY, porting to ..... 9

## D

DLL ..... 11

dll, getting ..... 11

documents, latest version ..... 3

## E

earlier netCDF versions ..... 3

extra\_test requirements ..... 3

extra\_test, running ..... 6

## F

ffio.c ..... 9

flex and yacc ..... 3

## G

GNU make ..... 9

## I

install directory ..... 5

installation requirements ..... 3

installing netCDF ..... 7

## L

large file tests ..... 6

large file tests requirements ..... 3

little endian ..... 9

## M

m4 ..... 3

## W

windows build ..... 11

mailing lists ..... 16

make extra\_test ..... 6

make install ..... 7

make test ..... 6

make, running ..... 6

makeinfo ..... 3

Microsoft ..... 11

## N

ncconfig.h ..... 9

ncconfig.in ..... 9

ncconfig.inc ..... 9

ncdump, windows location ..... 11

ncgen, windows location ..... 11

ncio ..... 9

ncx.m4 ..... 9

NET ..... 11

netcdf.dll, location ..... 11

netcdf.lib ..... 11

nm ..... 3

## P

porting notes, additional ..... 9

posixio.c ..... 9

prefix argument of configure ..... 5

problems, reporting ..... 16

## Q

quick unix instructions ..... 1

## R

reporting problems ..... 16

running configure ..... 5

running make ..... 6

## S

SunOS 64-bit build ..... 5

## T

tests, running ..... 6

troubleshooting ..... 15

turning off C++, Fortran interface ..... 15

## V

VC++ ..... 11

VC++ 6.0, building with ..... 12

VC++ 6.0, using netcdf with ..... 13

VC++.NET, building with ..... 14

VC++.NET, using netcdf with ..... 14

visual studio 2003 properties ..... 14

