# Wind-US Installation Guide*

The NPARC Alliance

*NASA Glenn Research Center*
*Cleveland, Ohio*

*USAF Arnold Engineering Development Center*
*Tullahoma, Tennessee*

*This is an unnumbered version of this document, created November 4, 2016. Please send corrections, additions, ideas, etc., to Dennis Yoder at *Dennis.A.Yoder@nasa.gov*.

# Contents


**1 Overview** **1**

**2 Obtaining Wind-US and the Tools** **3**
2.1 IVMS (Internet Version Management System) . . . 3
2.2 Wind Projects and Versions . . . 3
2.2.1 Wind-US Application Distribution . . . 5
2.2.2 Wind-US Build Distribution . . . 6
2.3 Tools Projects . . . 7
2.3.1 Tools Application Distribution . . . 8
2.3.2 Tools Build Distribution . . . 10
2.3.3 Individual Tools . . . 10

**3 Installing the Application Distributions** **11**
3.1 Installing on a UNIX System . . . 11
3.1.1 Using Symbolic Links for Similar Systems . . . 13
3.1.2 Installation for NFS Access . . . 15
3.1.3 Installation for Parallel Execution on Distributed Systems . . . 15
3.2 Installing on a Windows System . . . 16

**4 Installing the Build Distributions** **17**
4.1 Building Wind-US . . . 17
4.2 Building the Tools . . . 20

**5 Installing and Running the Build Distributions on the NAS** **23**
5.1 Security Issues on the NAS . . . 23
5.2 Storage Issues on the NAS . . . 23
5.3 Computational Resources on the NAS . . . 23
5.4 Building Wind-US on the NAS . . . 24
5.5 Building the Tools on the NAS . . . 28
5.6 Running Wind-US on the NAS . . . 34

**6 Porting Wind-US to a New UNIX Platform** **37**
6.1 Porting Guidelines . . . 37
6.2 Frequently (or not) Encountered Problems . . . 38

# 1 Overview

The NPARC Alliance flow simulation system is distributed to users as a collection of compressed tar files, containing Wind-US and several pre- and post-processing tools. Both Wind-US and the tools are available in two different "distributions" — an *application* distribution and a *build* distribution. The *application* distributions are intended for those who will be running the codes, but will not be modifying them. The *build* distributions are intended for those who must build the executables themselves, either to run on a computational platform not currently supported by the NPARC Alliance, or to include some compile-time option not present in the application distribution.[1]

*Wind-US Application Distribution*

The Wind-US *application* distribution includes all the executable programs and scripts necessary to run Wind-US on a supported computational platform. Due to limited resources, pre-compiled Wind-US 2.0 executables are currently only available for 32-bit and 64-bit Linux systems.

The application distribution includes PVM but not MPI, and the Wind-US executable was built assuming that PVM message passing will be used for parallel runs. If Wind-US is to be run on a multi-processor system using MPI, you'll need to build the Wind-US executable yourself using the *build* distribution, and link with the locally-installed MPI libraries.

*Tools Application Distribution*

A tools *application* distribution includes all the executable programs and scripts necessary to run one or more of the various pre- and post-processing tools on a supported computational platform. This includes:

- GMAN, a pre-processor used for setting boundary conditions and multi-zone connectivity (see the *GMAN User's Guide*)
- MADCAP, currently used primarily as a successor to GMAN, and when fully implemented will allow access under one user interface to the full range of tools and processes used to perform CFD analyses (see the *MADCAP User's Guide*)
- CFPOST, a post-processor used to list and plot results, generate reports, and produce files for other post-processors (see the *CFPOST User's Guide*
- A variety of smaller utilities, described in the *Wind-US Utilities Guide*

Pre-compiled tools executables are generally available for the same platforms as Wind-US.

*Wind-US Build Distribution*

The Wind-US *build* distribution contains all the files needed to build and install Wind-US on a variety of platforms. This includes:

- Source code for the Wind-US flow solver
- Source code for all the required library routines, including PVM
- Scripts needed to run Wind-US
- Makefiles for a variety of computational platforms

*Tools Build Distribution*

A tools *build* distribution contains all the files needed to build and install one or more of the tools on a variety of platforms. This includes:

[1] Developers also have access to a *development* distribution. This is discussed in the *Development Environment* section of the *Wind-US Developer's Reference*, accessible only to registered Wind-US developers.

- Source code for the tool(s)
- Source code for all the required library routines
- Scripts needed to run the tool(s)
- Makefiles for a variety of computational platforms

# 2 Obtaining Wind-US and the Tools

Wind-US, and the associated pre- and post-processing tools, are releasable to all U. S. owned companies, public and private universities, and government agencies. However, only U. S. citizens and resident aliens may have access to the software. In general, requests from foreign owned, controlled, or influenced (i.e., those with nonresident foreign nationals on the board of directors) corporations will not be granted. Instructions for becoming a registered user may be accessed from the NPARC Alliance home page at http://www.grc.nasa.gov/WWW/wind/index.html. You may also contact the NPARC Alliance User Support Team via email to `nparc-support@arnold.af.mil`, or by phone at (931) 454-7885.

## 2.1 IVMS (Internet Version Management System)

Approved users download Wind-US and the tools over the World-Wide Web using the IVMS (Internet Version Management System). Instructions on obtaining an IVMS account will be supplied by the NPARC Alliance User Support Team when you become a registered Wind-US user. After your IVMS account has been activated, you can login to IVMS through the IVMS login page.

After logging in to IVMS, you'll initially be connected to an opening page similar to the one shown in Figure 1.

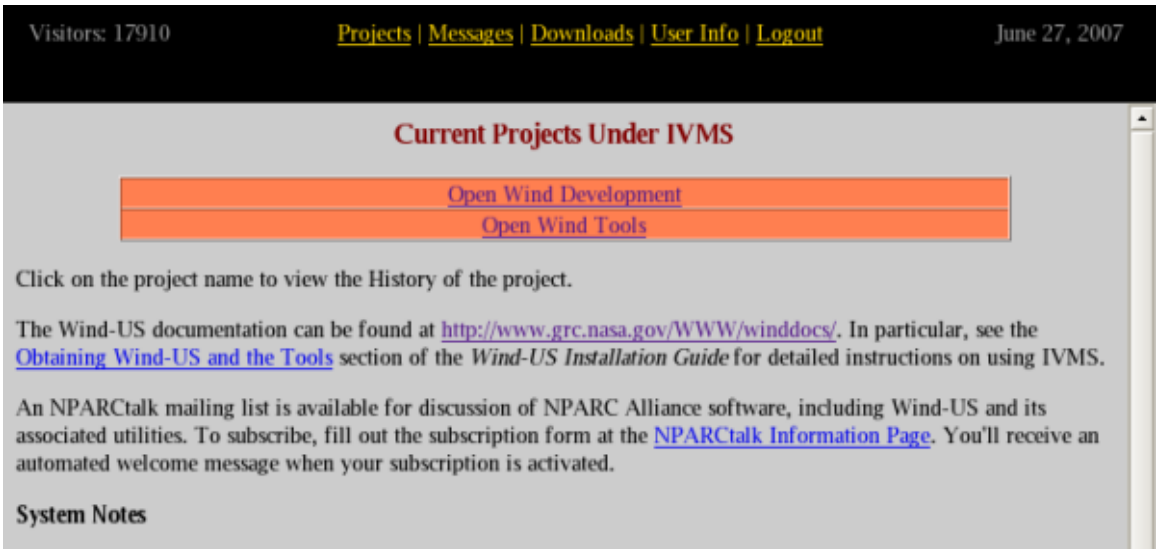


**Figure 1:** IVMS Projects Page.

Clicking on "Open Wind Development" or "Open Wind Tools" brings up a page listing the various Wind or Tools "Projects" that are currently available.

## 2.2 Wind Projects and Versions

Several different versions of Wind and Wind-US may be listed on the Wind Projects Page, shown in Figure 2.

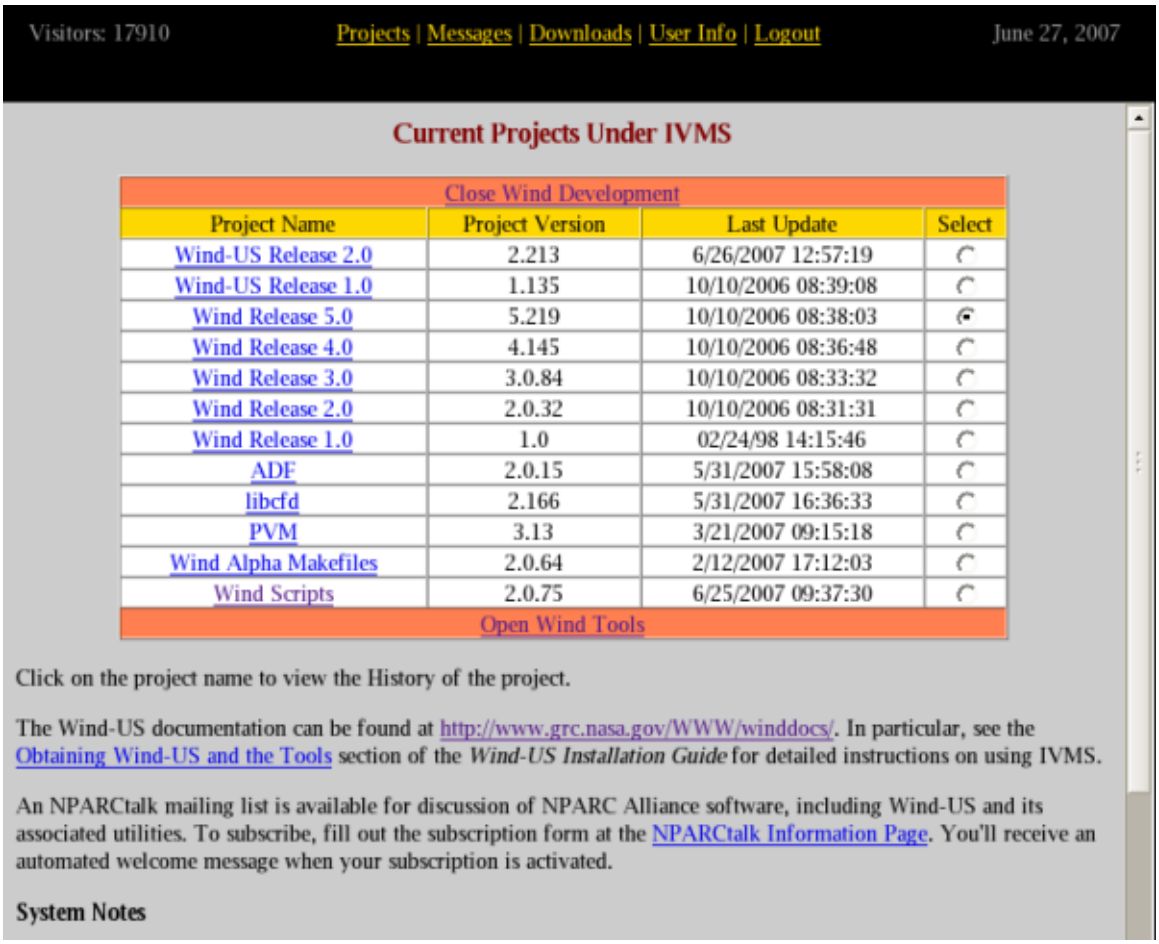


**Figure 2:** Wind Projects Page.

Officially-released versions are listed as "Wind Version $n$" or "Wind-US Version $n$," where $n$ is the version number. The most recent officially-released version is called the *production* version. As new releases are made, older released versions may also become available.

Once an official release is made, the production version is frozen except for bug fixes, and no new features are added. New feature development is done using the *alpha* version, the current "working" version of the code. About 2–3 months before the next scheduled release date, a *beta* version may be listed, and is intended as the next production version. Note, though, that executables for the beta and alpha versions will usually not be available on as many computational platforms as the production version.

The project of interest to a typical Wind-US code user is "Wind-US Release 2.0," containing the *application* and *build* distributions for the current production version of Wind-US. (See Section 1 for a description of the different "distributions".)

The version number and date for the various Wind projects are shown on the Wind Projects page, in the columns labeled "Project Version" and "Last Update". Note that the version number is for the source code, not necessarily the executables. If bug fixes have been made to the production version, for example, the executable in the application distribution may be slightly out of date.

The executables for the various supported platforms are periodically, but irregularly, updated. The version numbers and dates for the executables are shown in a table at the start of the "Source History" page, viewable by clicking on the project name in the first column of the Wind Projects page. A typical table is shown below.

| Project Name | Current Version | Last Update |
|---|---|---|
| Wind-US Release 2.0 | 2.213 | 6/26/2007 12:57:19 |
| LINUX32-GLIBC2.3/XEON | 2.212 | 6/25/2007 11:05:41 |
| LINUX64-GLIBC2.3/OPTERON | 2.212 | 6/25/2007 11:05:41 |

From the table, the current version of the source code for Wind-US Release 2.0 is 2.213. The "Last Update" date shown for Wind-US Release 2.0 is either the date the source code was last updated, or the most recent date an executable was updated. Again from the above table, the 32-bit executable for Linux Xeon systems was created from Wind-US version 2.212, on June 25, 2007. Note that the latest executable isn't normally the same for all platform/CPU types.

The changes made to the code since its release are described in the "Source History" page, following the above table. By examining the information in the "Source History" page, you can determine whether or not the currently available executable is suitable for your particular problem. If a change has been made that you need, but is not included in the currently available executable, you can build the executable yourself by downloading and installing the build distribution.

### 2.2.1 Wind-US Application Distribution

To download the Wind-US application distribution, select "Wind-US Release 2.0" on the Wind Projects Page (see Figure 2) by clicking on the button in the right-hand column, then click on "Downloads" in the menu at the top of the page. You'll be connected to the Wind-US Download Page, like the one shown in Figure 3 for Dr. Wyn D. Ooser. The top part of this page, shown in the figure, deals with downloading the application distribution, i.e., the scripts and pre-compiled Wind-US executable. The bottom part deals with downloading the build distribution, and will be discussed in the next section.

Select the desired computer and operating system by clicking on the appropriate entry in the "Machine OS" list, and the desired CPU type by clicking on the appropriate entry in the "CPU" list. Executables for multiple platforms and/or CPU types must be downloaded separately.

Note that the "Machine OS" and "CPU" types need not (necessarily) exactly match your system. For example, the LINUX32-GLIBC2_3/XEON executable will probably run on any Linux system with an x86-based CPU and with glibc version 2.3 installed. Symbolic links can be used in this situation to "point to" the executable. See Section 3.1 for more details.

Next, select (or de-select) the appropriate entries from the list located just above the "Download" button. For a UNIX (or Linux) system, new users should select the Wind-US executable, the PVM run-time scripts, and the UNIX Wind-US scripts. Note that the UNIX scripts are not platform-dependent, so when downloading executables for additional UNIX platforms and/or CPU types, only the Wind-US executable need be selected.

After selecting the Machine/OS and CPU, and the items to download, click the "Download" button. IVMS will package the selected items into a gzip'ed tar file, and prompt you for a file name, which should end with the extension *.tar.gz* for UNIX systems. The whole process may take several minutes, depending on the size of the items being downloaded and the load on the IVMS server, so please be patient.

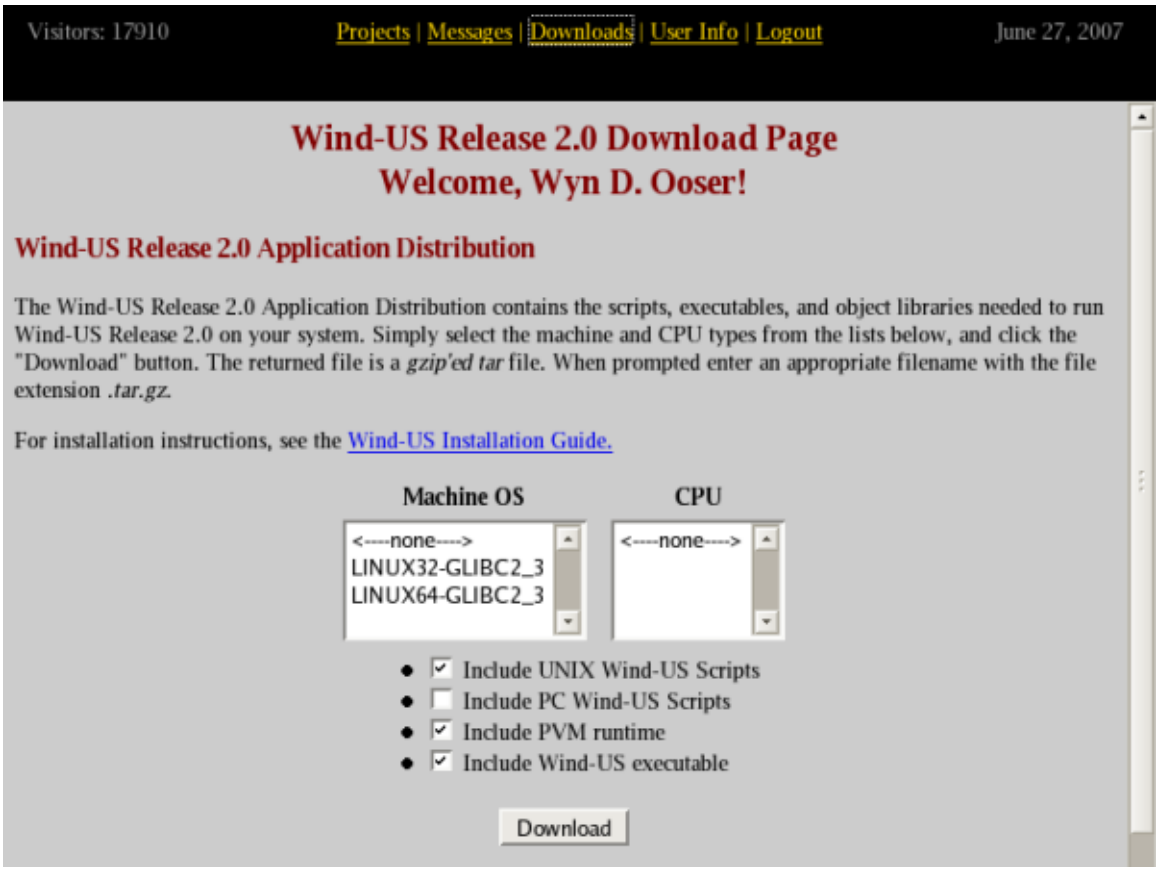


**Figure 3:** Wind-US Download Page, Application Distribution.

### 2.2.2 Wind-US Build Distribution

If a pre-compiled Wind-US executable is not available for the platform you'll be running on, or if you need a more recent version than is currently available, you'll need to download the Wind-US build distribution.

To download the Wind-US build distribution, simply click on the "Download" button in the bottom part of the Wind-US Download Page, shown in Figure 4. IVMS will package the necessary files into a gzip'ed tar file, and, if the "Package files for downloading later" button is not selected, you'll be prompted for the file name, which should end with the extension *.tar.gz*. The whole process may take several minutes, depending on your Internet connection and the load on the IVMS server, so please be patient.

Under some conditions, this process may time out before completion, resulting in a "Document contains no data" error. The "Downloading later" option was added to solve this problem. If the "Package files for downloading later" button is selected, IVMS will package the necessary files into a gzip'ed tar file, and a "Get it here" link will appear in the IVMS "System Messages" window. Click on it, and you'll be prompted for the file name, and the process will continue as normal. You still may get a "Document contains no data" error, but it may be ignored.

Occasionally, the "System Messages" window fails to refresh properly. If the "Get it here" link fails to appear after a few minutes, manually reload the "System Messages" window. In Netscape,


**Figure 4:** Wind-US Download Page, Build Distribution.

this is done using a right-mouse click in the window, and selecting the "Reload" menu option. Some users have also reported that clicking on the "Get it here" link occasionally results in them being returned to the IVMS login screen. The cause of this problem is unknown.

## 2.3 Tools Projects

To download the various pre- and post-processing tools, first click on "Open Wind Tools" on the IVMS opening page. That will display the Tools Projects Page, shown in Figure 5. Tools may be downloaded as a group, either as executables in a "tools application distribution" or as source code in a "tools build distribution". Application and/or build distributions may also be downloaded for individual tools. However, for the application distribution, it's recommended that at least some tools be downloaded as a group, which will include a tools installation script. The installation script is not included when an application distribution is downloaded for an individual tool.

Note that at least one of the tools, either GMAN (see the *GMAN User's Guide*) or MADCAP (see the *MADCAP User's Guide*) must be downloaded, since it's required for setting boundary condition types for input to Wind-US. Another, CFPOST (see the *CFPOST User's Guide*), is very useful in examining the results from Wind-US runs. Other tools are described in the *Wind-US Utilities Guide*.

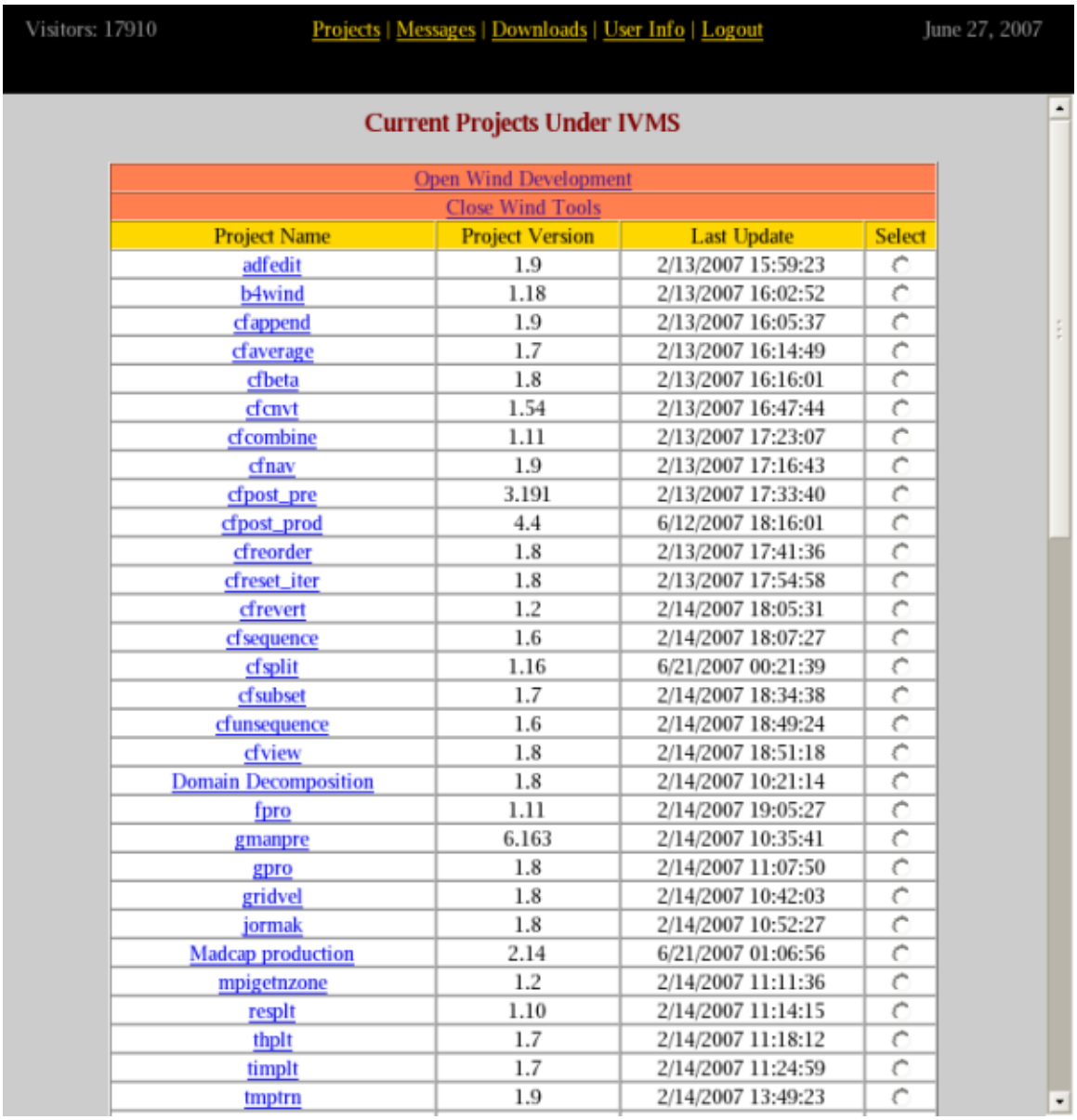


**Figure 5:** Tools Projects Page.

### 2.3.1 Tools Application Distribution

To download a tools application distribution for a group of tools, on the Tools Projects Page (Figure 5) select "Tools Makefiles" by clicking on its button in the right-hand column near the bottom of the page, then click on "Downloads" in the menu at the top of the page. You'll be connected to the Tools Makefiles Download Page, the top part of which is shown in Figure 6. The bottom part of the page, not shown in the figure, deals with downloading the build distribution, and will be discussed in the next section.

The executables in the tools distribution are available for a variety of systems. Select the desired


**Figure 6:** Tools Makefiles Download Page.

computer and operating system by clicking on the appropriate entry in the "Machine OS" list, and the desired CPU type by clicking on the appropriate entry in the "CPU" list. Tools distributions for multiple platforms and/or CPU types must be downloaded separately.

The tools to be included in the tools application distribution are chosen by clicking on the buttons to the left of the tool names. Note that all the tools are selected by default, except for CFPOST (cfpost_pre in the list), GMAN (gmanpre), and MADCAP (Madcap production). These are significantly larger than the other tools, and users with slower Internet connections may want to download them separately from the Tools Projects Page.

After selecting the Machine/OS and CPU, and the tools to download, click the "Download" button. IVMS will package the necessary files into a gzip'ed tar file, and prompt you for a file name, which should end with the extension *.tar.gz* for UNIX systems. As with the other distributions, the whole process may take several minutes, so please be patient.

### 2.3.2 Tools Build Distribution

If pre-compiled tools are not available for the platform you'll be running on, or if you need more recent versions than are currently available, you'll need to download the tools build distribution.

To download the tools build distribution, simply click on the "Download" button at the bottom of the Build Distribution section of the Tools Makefiles Download Page. IVMS will package the necessary files into a gzip'ed tar file, and, if the "Package files for downloading later" button is not selected, you'll be prompted for the file name, which should end with the extension *.tar.gz*. The whole process may take several minutes, depending on your Internet connection and the load on the IVMS server, so please be patient.

Note that the procedure described in the previous section for selecting which tools to include in the application distribution *does not apply* to the build distribution. The build distribution will automatically include all the tools except for GMAN, CFPOST, and MADCAP. If it's necessary to build GMAN, CFPOST, or MADCAP, the build distribution for each of them must be downloaded separately.

### 2.3.3 Individual Tools

The procedure for downloading an application or build distribution for an individual tool is essentially the same as described previously for the application and build distributions for a group of tools. On the Tools Projects Page (see Figure 5) select the desired tool by clicking on the button for that tool in the right-hand column, then click on "Downloads" in the menu at the top of the page. You'll be connected to the Download Page for the selected tool.

To download the application distribution for that tool (i.e., the executable), select the desired computer and operating system as usual, and then click the "Download" button. IVMS will package the necessary files into a gzip'ed tar file, and prompt you for a file name, which should end with the extension *.tar.gz* for UNIX systems.

To download the build distribution, simply click on the "Download" button at the bottom of the Build Distribution section of the Download Page for that tool. IVMS will package the necessary files into a gzip'ed tar file, and, if the "download later" option is not used, you'll be prompted for the file name, which should end with the extension *.tar.gz*.

# 3 Installing the Application Distributions

## 3.1 Installing on a UNIX System

After downloading the Wind-US and tools application distributions for the platforms to be used, the recommended procedure to unpack and install them is as follows:

1. Put the gzip'ed tar files containing the Wind-US and tools application distributions into an appropriate directory, such as *wind_download*. This directory does not have to be the permanent parent directory for the scripts and executables, but it can be.

2. In that directory, unpack the files by doing, for each file:

   `gunzip -c` *filename* `| tar xvf -`

   where *filename* is the file name, including the *.tar.gz* extension. Note that if application distributions were downloaded for several different machines, all of the gzip'ed tar files should be located in the same directory, and unpacked before proceeding to the actual installation. Similarly, if application distributions for some tools have been downloaded individually, it's recommended that they also be unpacked before continuing. This will allow all the tools to be installed in one step, rather than separately.

3. Install Wind-US first, by running the *Install.appl* script in the directory containing the application distributions. When prompted, enter the parent directory where Wind-US is to be installed. A *wind* subdirectory will be created directly below the directory you specify, and all files will be installed in subdirectories below that. E.g., if you specify */usr/local* as the parent installation directory, the result will be a directory structure in */usr/local/wind*.

   The *Install.appl* script does not change any of your system or user files. It simply copies the executables and scripts to the appropriate locations below the directory you specify, and modifies the *cfd.login* and *cfd.profile* files in the application distribution accordingly.

4. Next, to finish the installation of Wind-US, *csh* and *tcsh* users must add the following line to their *.login* file in their home directory.

   `source` *install_directory*`/wind/bin/cfd.login`

   where *install_directory* is the full path name for the parent installation directory you specified in the previous step.

   Similarly, *sh*, *bash*, and *ksh* users must add the following line to their *.profile* file in their home directory.

   `.` *install_directory*`/wind/bin/cfd.profile`

   This will cause the contents of *cfd.login* (or *cfd.profile*) to be executed each time you log in, setting/modifying the following environment variables:

   - `CFDROOT`, specifying the root location of the Wind-US executable(s) and scripts (i.e., *install_directory*`/wind`)
   - `SYSTEM` and `SYSTEM_CPU`, defining the type of system and CPU you're using
   - `PATH`, to include the locations of the newly-installed executables and scripts in your search path

5. Log out and log back in, so that the *cfd.login* (or *cfd.profile*) script gets executed. (Or alternatively, issue the command "`source .login`" (or "`source .profile`") in your home directory.)

This is necessary in order to define the `CFDROOT`, `SYSTEM`, and `SYSTEM_CPU` environment variables, that the tools installation and usage also require.

6. Next install the tools, by running the *Install.tools* script in the directory containing the application distributions. This step is exactly analogous to the one above for Wind-US. When prompted, enter the parent directory where the tools are to be installed. While not strictly necessary, it's recommended that the tools be installed under the same parent directory as used for Wind-US. A *tools* subdirectory will be created directly below the directory you specify, and all files will be installed in subdirectories below that. E.g., if you specify */usr/local* as the parent installation directory, the result will be a directory structure in */usr/local/tools*.

   The *Install.tools* script does not change any of your system or user files. It simply copies the executables and scripts to the appropriate locations below the directory you specify, and modifies the *tools.login* and *tools.profile* files in the application distribution accordingly.

7. To finish the installation of the tools, *csh* and *tcsh* users must add the following line to their *.login* file in their home directory.

   `source` *install_directory*`/tools/bin/tools.login`

   where *install_directory* is the full path name for the parent installation directory you specified in the previous step.

   Similarly, *sh*, *bash*, and *ksh* users must add the following line to their *.profile* file in their home directory.

   `.` *install_directory*`/tools/bin/tools.profile`

   This will cause the contents of *tools.login* (or *tools.profile*) to be executed each time you log in, setting/modifying the following environment variables:

   - `TOOLSROOT`, specifying the root location of the tools executable(s) and scripts (i.e., *install_directory*`/tools`)
   - `PATH`, to include the locations of the newly-installed executables and scripts in your search path

8. Finally, log out and log back in again, so that the *tools.login* (or *tools.profile*) script gets executed (or alternatively, issue the command "`source .login`" (or "`source .profile`") in your home directory), and the necessary environment variables get defined.

After completing the above steps, the directory structure will be similar to the example shown below, where *wind_install* is the directory specified in steps 3 and 6. (This example assumes that 32-bit executables were installed for a Linux system with *glibc* 2.3 and an Intel Xeon processor, and SGI systems with R12000 and R14000 processors. The actual "system" and "CPU" directory names for your installation will correspond to the choices you made for "Machine OS" and "CPU" when you downloaded the application distribution(s), as described in Section 2.2.1.)

```
wind_install/
   wind/
      LINUX32-GLIBC2.3/
         XEON/
            bin/
               Wind-US executable
      SGI32-IRIX6.5/
         R12000/
            bin/
```

```
                Wind-US executable
            R14000/
               bin/
                  Wind-US executable
         bin/
            Wind-US and PVM scripts
            chemistry/
               Chemistry data (.chm) files
         pvm/
            LINUX32-GLIBC2.3/
               XEON/
                  PVM executables
            SGI32-IRIX6.5/
               R12000/
                  PVM executables
               R14000/
                  PVM executables
      tools/
         LINUX32-GLIBC2.3/
            XEON/
               bin/
                  Tools executable(s)
         SGI32-IRIX6.5/
            R12000/
               bin/
                  Tools executable(s)
            R14000/
               bin/
                  Tools executable(s)
         bin/
            Tools scripts
```

### 3.1.1 Using Symbolic Links for Similar Systems

As noted in Section 2.2.1, if an executable is not available for the exact type of system and CPU you need to run on, a symbolic link can often be set up to specify that a different (but compatible) executable should be used.

For example, most (all?) modern Linux distributions include *glibc* Version 2.3. Suppose your `SYSTEM` environment variable is equal to `LINUX32-RH9`, but only executables for `LINUX32-GLIBC2.3` are available. You can run the `LINUX32-GLIBC2.3` executables by making `LINUX32-RH9` a symbolic link to the directory `LINUX32-GLIBC2.3`. I.e., in the *wind*, *wind/pvm*, and *tools* directories, do

```
ln -s LINUX32-GLIBC2.3 LINUX32-RH9
```

The resulting directory structure will be (starting from the example shown in the previous section, here in abbreviated form):

```
wind_install/
   wind/
      LINUX32-GLIBC2.3/
         ...
```

```
      LINUX32-RH9 -> LINUX32-GLIBC2.3/
      SGI32-IRIX6.5/
         ...
      bin/
         ...
      pvm/
         LINUX32-GLIBC2.3/
            ...
         LINUX32-RH9 -> LINUX32-GLIBC2.3/
         SGI32-IRIX6.5/
            ...
   tools/
      LINUX32-GLIBC2.3/
         XEON/
            ...
      LINUX32-RH9 -> LINUX32-GLIBC2.3/
      SGI32-IRIX6.5/
         ...
      bin/
         ...
```

where the notation "`LINUX32-RH9 -> LINUX32-GLIBC2.3/`" is used to indicate that `LINUX32-RH9` is a symbolic link pointing to `LINUX32-GLIBC2.3/`.

Similarly, on SGI systems an executable built on an R14000 CPU will also run on an R16000 CPU, for example. In this case, to run the R14000 executables on R16000 systems, in the `SGI32-IRIX6.5` directories below *wind*, *wind/pvm*, and *tools* you would make `R16000` a symbolic link to the directory `R14000`.

```
ln -s R14000 R16000
```

The resulting directory structure will be (again starting from the example shown in the previous section, here in even more abbreviated form):

```
wind_install/
   wind/
      LINUX32-GLIBC2.3/
      SGI32-IRIX6.5/
         R12000/
         R14000/
         R16000 -> R14000/
      bin/
      pvm/
         SGI32-IRIX6.5/
            R12000/
            R14000/
            R16000 -> R14000/
   tools/
      LINUX32-GLIBC2.3/
      SGI32-IRIX6.5/
         R12000/
         R14000/
         R16000 -> R14000/
      bin/
```

### 3.1.2 Installation for NFS Access

If multiple users in an organization will be running Wind-US, it's convenient to install the Wind-US and tools executables below a parent directory that can be accessed by all the users via NFS. This way, the location and method of accessing the executables will be the same on all user machines, and they can be updated without impacting the user significantly. It is suggested that one individual from an organization download the needed application distributions and serve as a single point of contact for Wind-US.

When the scripts and executables are to be accessed via NFS, if the path name used by the user to access the NFS-mounted file system is different from the actual directory name on the "host" system, a slight variation of the above installation procedure is necessary. That's because the user's environment variables `CFDROOT` and `TOOLSROOT` must define the NFS-mounted location of the *wind* and *tools* subdirectories. Similarly, the lines users add to their *.login* or *.profile* files should specify path names used to access the NFS-mounted file system.

NFS file systems are typically accessed using *automount*. As an example, assume that Wind-US and the tools have been installed on a host workstation named *wind_machine*, and that the parent installation directory is */usr/local/wind_nfs*. On *wind_machine*, that directory is exported read-only to the "home" system, normally an individual's workstation, of each Wind-US user. The user can then access that directory via automount by adding "*/net/wind_machine*" to the beginning of the path name. E.g.,

```
cd /net/wind_machine/usr/local/wind_nfs
```

The *cfd.login*, etc., files on *wind_machine* were copied to new files named *cfd.nfs.login*, etc.[2] The definitions of `CFDROOT` and `TOOLSROOT` in *cfd.nfs.login*, etc., were modified to point to the automount'ed location of the *wind* and *tools* subdirectories. I.e., in *cfd.nfs.login* `CFDROOT` is defined as

```
setenv CFDROOT /net/wind_machine/usr/local/wind_nfs/wind
```

and in *tools.nfs.login* `TOOLSROOT` is defined as

```
setenv TOOLSROOT /net/wind_machine/usr/local/wind_nfs/tools
```

Finally, using the above example, *csh* and *tcsh* users accessing Wind-US via automount would modify their *.login* file to add the lines

```
source /net/wind_machine/usr/local/wind_nfs/wind/bin/cfd.nfs.login
source /net/wind_machine/usr/local/wind_nfs/tools/bin/tools.nfs.login
```

and *sh* and *ksh* users would modify their *.profile* file to add the lines

```
. /net/wind_machine/usr/local/wind_nfs/wind/bin/cfd.nfs.profile
. /net/wind_machine/usr/local/wind_nfs/tools/bin/tools.nfs.profile
```

### 3.1.3 Installation for Parallel Execution on Distributed Systems

It should be noted that Wind-US does not need to be installed on each individual machine in order to run in parallel mode using a network of distributed systems. The executables for all the different types of systems and CPUs being used are installed only on the master system. The Wind-US run scripts automatically take care of copying the appropriate Wind-US and PVM executables and files

[2] This was done so that users logging directly onto *wind_machine* could still access Wind-US as described in the previous section.

to the systems being used, starting and stopping the message passing software, and cleaning up at the end of the run. See the "Parallel Processing" section of the *Wind-US User's Guide* for details.

## 3.2 Installing on a Windows System

Unfortunately, due to resource limitations, the NPARC Alliance cannot officially support a Windows version of Wind-US. Nevertheless, application distributions for Windows may be available for Wind-US and some of the tools. These executables will generally not be as current as those in the supported Unix versions. The recommended installation procedure for Windows application distributions is described below.

After downloading the gzip'ed tar files containing the Wind-US and tools application distributions to some temporary directory:

1. Uncompress the gzip'ed tar file using some appropriate decompression software, such as WinZIP, extracting all the files into the temporary directory. With WinZIP this is done by clicking on the "Extract" button.
2. An "Install" link will appear in the temporary directory. Double click this to install the code(s). A DOS window will open.
3. Specify where you want to install the code(s) and a default name for the location for running applications. The applications directory is just the default and can be changed interactively later.
4. Follow the instructions regarding modification of the *autoexec.bat* file, then reboot.

An icon should appear on your desktop. Double-clicking on it will start execution of Wind-US.

# 4 Installing the Build Distributions

If pre-compiled executables for Wind-US and/or the tools are not available for the platform you'll be running on, you'll need to install the build distribution. This section describes the procedure for UNIX systems.

## 4.1 Building Wind-US

Makefiles are distributed with the Wind-US build distribution for a variety of platform, operating system, and CPU combinations, including:

- Convex
- Cray
- Hewlett-Packard
  - Exemplar CSPP; PA-8000 processor
  - HP-UX 11.*x*; PA-8600 processor
- IBM RS6000; PC600 processor
- Linux
  - *glibc* 2.3; Athlon, Xeon, Opteron, and X86 processors
  - SUSE; Opteron processor
- Silicon Graphics IRIX 6.5; R4400, R5000, R8000, R10000, R12000, R14000, R16000 processors
- Sun Solaris 8 and 9; SPARC4U processor

If you need to build the code for a platform not listed above, you'll need to first create the appropriate makefiles, using the existing ones as a starting point. See Section 6 for further information. The NPARC Alliance User Support Team (`nparc-support@arnold.af.mil`, or (931) 454-7885) may also be able to provide guidance when creating and naming new makefiles.

Assuming the appropriate makefiles exist, after downloading the gzip'ed tar files, the recommended procedure is described below. Note that steps 4 through 6 can be avoided by installing a Wind-US application distribution (even if it doesn't include the pre-compiled executable), as described in Section 3. The Wind-US build distribution is a complete package, however, and, using the following procedure, a working system can be constructed from it alone.

1. Put the gzip'ed tar file containing the Wind-US build distribution into an appropriate directory.
2. In that directory, unpack the file by doing:

   `gunzip -c` *filename* `| tar xvf -`

   where *`filename`* is the file name, including the *.tar.gz* extension. This creates a directory named *wind-dev* containing the Wind-US source code.
3. Check to see if the `CFDROOT`, `SYSTEM`, and `SYSTEM_CPU` environment variables have been set to appropriate values, by doing

   ```
   printenv CFDROOT
   printenv SYSTEM
   printenv SYSTEM_CPU
   ```

   If all three are correctly set, skip ahead to step 7.
4. If you're using the *csh* or *tcsh* shell, edit the file *dir_name/wind-dev/bin/cfd.login*, where *dir_name* is the name of the directory used in step 1, to modify the definition of `CFDROOT` as follows:

```
if (! "$?CFDROOT") then
#=BEGROOT=
  setenv CFDROOT "path_name/wind-dev"
#=ENDROOT=
endif
```

where *path_name* is the full path name (starting with a "/") of the directory used in step 1.

Similarly, *bash*, *ksh*, or *sh* users would edit the file *dir_name/wind-dev/bin/cfd.profile* to modify the definition of `CFDROOT` to:

```
if [ ! "$CFDROOT" ] ; then
#=BEGROOT=
  CFDROOT="path_name/wind-dev"
#=ENDROOT=
  export CFDROOT
fi
```

5. Next, *csh* and *tcsh* users must edit the *.login* file in their home directory and add the following line:

   `source` *path_name*`/wind-dev/bin/cfd.login`

   where again *path_name* is the full path name of the directory used in step 1. Similarly, *bash*, *ksh*, and *sh* users must edit their *.profile* file and add the following line:

   `.` *path_name*`/wind-dev/bin/cfd.profile`

   This causes the contents of the *cfd.login* (or *cfd.profile*) file to be executed automatically at login time to set up the environment variables `CFDROOT`, `SYSTEM`, and `SYSTEM_CPU`, that are required for installing and running Wind-US, and to modify your `PATH` to include the location of the Wind-US executable.

6. At this point, log out and log back in to execute your *.login* or *.profile* file (depending on your login shell). Alternatively, in your home directory execute (for *csh* and *tcsh* users)

   ```
   source .login
   ```

   or (for *sh* and *ksh* users)

   ```
   . .profile
   ```

   Either way, return to step 3.

7. Define the environment variable `WIND_DEV` as the location of the directory containing the top-level Makefile. If in step 1 the Wind-US build distribution was unpacked in the directory */usr/local/wind*, for example, *csh* and *tcsh* users would do

   ```
   setenv WIND_DEV /usr/local/wind/wind-dev
   ```

8. *cd* into the *wind-dev* directory

   ```
   cd wind-dev
   ```

9. The default values in the configuration files should be sufficient to produce an executable on most well-configured systems (assuming that compilers are installed). The primary exception to this is Linux, which does not have a "default" Fortran 90/95 compiler. Another potential problem is the location of the MPI libraries, which needs to be specified if you wish to use MPI for parallel runs. To be safe, review the contents of the following files or parts of files,

where *SYSTEM* and *SYSTEM_CPU* correspond to the `SYSTEM` and `SYSTEM_CPU` environment variables, paying special attention to the items noted below.

- *Makefile.configure*
  - If you're compiling on a multi-processor system with MPI message passing available, change the definition of `USE_MPI` from `NO` to `YES`. This will link in the MPI libraries, allowing the use of MPI message passing for parallel processing. Note, though, that because Wind-US uses dynamically-linked libraries, an executable created with `USE_MPI` equal to `YES` will not run on multi-processor systems without MPI.
  - If you'll never be running parallel jobs using PVM message passing, change the definition of `USE_PVM` from `YES` to `NO`.
  - The value of `PRECISION` may be changed to make all Fortran variables single or double precision. The default is a mix, with single precision for the mean flow solution and boundary conditions, and double precision for chemistry data, right-hand-side data, grid coordinates, and grid metrics.
- *source/Makefile.user*
- *source/makefiles/Makefile.include.SYSTEM.SYSTEM_CPU.xxx*, where *xxx* corresponds to *opt*, *dbx*, *pure*, or *check*.
  - Check the definition of `CPP`, which is the full path name for the C pre-processor *cpp*, to make sure it is correct for your system. You can locate *cpp* using the *whereis* command, at least on SGI and Sun systems.

    ```
    whereis -b cpp
    ```

  - For Linux systems, you will most likely have to change the definitions related to the Fortran and C compilers being used (i.e., the variables `ABI`, `FC`, `F90`, `LD`, etc.) These lines should be commented/uncommented as needed for your system, but additional changes may also be necessary or desired.
  - If you're compiling on a multi-processor system with MPI message passing, check the definition of `MPILIBS`, and modify it if necessary to use a different version or a non-standard location.
  - If your system doesn't support the Fortran long integer data type, add "`-DNOFLONG`" (without the quotes) to the definition of `WINDDEFS`. Errors in the compilation of *mem_management_module.f90* about an ambiguous definition for the generic interfaces `alloc` and `dealloc`, involving the specific interfaces for `allockindlong_1` and `allockindint_1`, etc., are an indication that "`-DNOFLONG`" is needed.

10. From the *wind-dev* directory, run *make* to create all the required libraries and executables for running Wind-US. To capture the output from *make* in the file *make.log* for later examination if something goes wrong, in addition to displaying it at the terminal, *csh* and *tcsh* users should do

    ```
    make opt |& tee make.log
    ```

    and *sh* and *ksh* users should do

    ```
    make opt 2>&1 | tee make.log
    ```

11. If you have a previous installation of Wind-US that you wish to update (i.e., `CFDROOT` points to some other directory than the one you are building from), you will need to install the Wind-US amd PVM executables. To do that, issue the commands

    ```
    make install
    make install_scripts
    make copy_pvm
    ```

    The first two commands copy the Wind-US executable to *$CFDROOT/SYSTEM/CPU/bin*, and the scripts to *$CFDROOT/bin*. The third copies the PVM executables and libraries to *$CFDROOT/pvm/lib/SYSTEM/CPU*, some PVM include files to *$CFDROOT/pvm/include*, and PVM scripts to *$CFDROOT/pvm/lib*. The parameters *SYSTEM* and *CPU* correspond to the `SYSTEM` and `SYSTEM_CPU` environment variables, respectively.

12. If this is a new installation, it would probably be best to log out and log back in before running Wind-US. This executes the shell start-up scripts, modifying the `PATH` environment variable to include the newly-created location for the Wind-US executable.

## 4.2 Building the Tools

Makefiles are distributed with the tools build distributions for many of the same platforms supported by the Wind-US build distribution. Unlike the Wind-US build distribution, in order to obtain the source for all the tools several downloads are required. The smaller tools are all bundled together and may be acquired from the "Downloads" page of the "Tools Makefiles" project. GMAN, CFPOST, and MADCAP are normally downloaded separately from their respective "Downloads" pages. Note that the Project Names for these are "gmanpre", "cfpost_pre", and "Madcap production", respectively.

Each build distribution is designed to be a completely independent package, so that the tools can be built without requiring any additional files from IVMS.[3] Thus, one could have separate directory trees for the Wind-US build distribution and each of the tools build distributions. This would lead to a great deal of duplication, however. Therefore, the build distributions are designed to overlay one another. The following instructions assume that all the tools are being built in the same tree.

1. Place all the source packages in the same directory. If you have already built Wind-US, the directory above *wind-dev* is the best place, in order to minimize duplication.

2. Unpack the archives for the tools you wish to compile by doing:

    ```
    gunzip -c filename | tar xvf -
    ```

    This places the source for the tools in directories under *wind-dev/tools-dev*.

3. Verify that the `CFDROOT`, `SYSTEM`, and `SYSTEM_CPU` environment variables, are correctly defined, as described in steps 3–6 in Section 4.1.

4. Check to see if the `TOOLSROOT` environment variable is properly set, by doing:

    ```
    printenv TOOLSROOT
    ```

    If it is, skip ahead to step 8.

5. If you're using the *csh* or *tcsh* shell, edit the file *dir_name/wind-dev/tools-dev/bin/tools.login*, where *dir_name* is the name of the directory used in step 1, to modify the definition of `TOOLSROOT` as follows:

[3] There are some exceptions to this, such as CFPOST, described below in step 10.

```
if (! "$?TOOLSROOT") then
#=BEGROOT=
  setenv TOOLSROOT "path_name/wind-dev/tools-dev"
#=ENDROOT=
endif
```

where *path_name* is the full path name (starting with a "/") of the directory used in step 1.

Similarly, *bash*, *ksh*, or *sh* users would edit the file *dir_name/wind-dev/tools-dev/bin/tools.profile* to modify the definition of `TOOLSROOT` to:

```
if [ ! "$TOOLSROOT" ] ; then
#=BEGROOT=
  TOOLSROOT="path_name/wind-dev/tools-dev"
#=ENDROOT=
  export TOOLSROOT
fi
```

6. Next, *csh* and *tcsh* users must edit the *.login* file in their home directory and add the following line:

   ```
   source path_name/wind-dev/tools-dev/bin/tools.login
   ```

   where again *path_name* is the full path name of the directory used in step 1. Similarly, *bash*, *ksh*, and *sh* users must edit their *.profile* file and add the following line:

   ```
   . path_name/wind-dev/tools-dev/bin/tools.profile
   ```

   This causes the contents of the *tools.login* (or *tools.profile*) file to be executed automatically at login time to set up the environment variable `TOOLSROOT`, and to modify your `PATH` to include the location of the tools executables.

7. At this point, log out and log back in to execute your *.login* or *.profile* file (depending on your login shell). Alternatively, in your home directory execute (for *csh* and *tcsh* users)

   ```
   source .login
   ```

   or (for *sh* and *ksh* users)

   ```
   . .profile
   ```

   Either way, return to step 4.

8. Define the environment variable `WIND_DEV` as the location of the directory containing the top-level Makefile. If in step 1 the tools build distributions were unpacked in the directory */usr/local/wind*, for example, *csh* and *tcsh* users would do

   ```
   setenv WIND_DEV /usr/local/wind/wind-dev
   ```

9. Most systems other than Linux should be able to use the default values in the configuration files without problem. On Linux systems, it is likely that the definitions related to the Fortran and C compilers will have to be changed. To be safe, check the following files to be sure they contain information that is appropriate for your system. (See the notes for step 9 in Section 4.1.)

   - *Makefile.configure*
   - *source/makefiles/Makefile.include.SYSTEM.SYSTEM_CPU.opt*
   - *source/makefiles/pvm_conf/SYSTEM.SYSTEM_CPU.def.opt*

- *tools-dev/Makefile*
- *tools-dev/Makedefs.SYSTEM*
- If GMAN is being built: *tools-dev/gmanpre/Makedefs.SYSTEM*
- If CFPOST is being built: *tools-dev/cfpost_pre/Makedefs.SYSTEM*
- If MADCAP is being built: *tools-dev/libmadcap/Makedefs.SYSTEM* and *tools-dev/madcapprod/Makedefs.SYSTEM*
- If GMAN, CFPOST, or MADCAP is being built: *tools-dev/libmdgl/SYSTEM.mkf*

10. In the *wind-dev* directory, run *make* to create all the required libraries and executables for all the tools for which you have installed source. To capture the output from *make* in the file *make_tools.log* for later examination if something goes wrong, in addition to displaying it at the terminal, *csh* and *tcsh* users should do

    ```
    make all_tools |& tee make_tools.log
    ```

    and *sh* and *ksh* users should do

    ```
    make all_tools 2>&1 | tee make_tools.log
    ```

    You can also compile tools individually, by doing

    `make` *tool_name*

    where *tool_name* is the name of the tool. Note that the names to be used for GMAN, CFPOST, and MADCAP are *gmanpre*, *cfpost_pre*, and *madcapprod*, respectively.

    If CFPOST is being built individually, the MADCAP library must be present, either as source code or as a previously-compiled object library. This can be satisfied in a couple of different ways.

    - If MADCAP is also to be built, unpack it in step 2 at the same time as the archives for the other tools that are being built.
    - If MADCAP is not being built, download the MADCAP library source code from IVMS (the Project Name is "Madcap production Library"), and unpack it, in the same directory as the rest of the tools, before building CFPOST.

11. If you have a previous installation of the tools that you wish to update (i.e., `TOOLSROOT` points to some other directory than the one you are building from), you will need to install the tools executables by doing

    ```
    make install_tools
    ```

    This copies the tools executables to *$TOOLSROOT/SYSTEM/CPU/bin*, where *SYSTEM* and *CPU* correspond to the `SYSTEM` and `SYSTEM_CPU` environment variables, as described in step 11 in Section 4.1.

12. If this is a new installation, it would probably be best to log out and log back in again before running any of the tools. This executes the shell start-up scripts, modifying the `PATH` environment variable to include the newly-created location for the tools executables.

Note that the build procedure for the tools is somewhat less mature than the Wind-US build process. Please report problems to the NPARC Alliance support team at *nparc-support@arnold.af.mil* or (931) 454-7885.

# 5 Installing and Running the Build Distributions on the NAS

Pre-compiled executables for Wind-US and/or the tools are not available for the NASA Advanced Supercomputing (NAS) systems: columbia and pleiades. User's must install the build distribution into their local directory. This section describes the procedure for doing this.

## 5.1 Security Issues on the NAS

User's are reminded that they are responsible for protecting the dissemination of Wind-US, particularly on a shared computer resource like the NAS. It is therefore recommended that users restrict the access permissions on their NAS home and nobackup directories to prevent access from other users. If this is not already the default behavior, one can use the following commands.

- To restrict access to an existing file or directory use the command:
  `chmod go-rwx` *filename*
- To restrict access to all future files and directories created during the current session use the command:
  `umask 077`
- Depending on which linux shell is being used, the `umask` command above can be placed in the *$HOME/.login* or *$HOME/.profile* login scripts to apply to future sessions.

## 5.2 Storage Issues on the NAS

Storage space on NAS is split between a rather limited (8 GB) *$HOME* directory and a larger (200 GB) */nobackup/$USER* directory. Most users install Wind-US into their home directory and submit jobs from their nobackup directory. Offline tape storage is available by transferring files to the machine called lou. Please see the NAS website (http://www.nas.nasa.gov/) for additional details.

## 5.3 Computational Resources on the NAS

The NAS computing cluster is comprised of various computing nodes, each of which contains a different number and type of core processors. The user can select which nodes to use by specifying the model name within the PBS script. The default option for most queues is Westmere.

Because each processor type has a different computational efficiency, NAS charges for their use via a Standard Billing Unit (SBU). In this scheme, use of faster computing nodes incurs a larger SBU cost. Each submitted job is is given exclusive access to the requested nodes. The user is charged for using each node, even if the job does not utilize all of the available processors. To make the most of their allotted time on NAS, users should try to fully utlize each computing node. The table below summarizes the available computing resources.

For more information, visit:

- http://www.nas.nasa.gov/hecc/support/kb/Resources-Request-Examples_188.html
- http://www.nas.nasa.gov/hecc/support/kb/Pleiades-Configuration-Details_77.html

**Table 1: NAS Computing Resources**

| Processor Type | Model Name | SBU/node | CPUs/node | RAM(GB)/node |
|---|---|---|---|---|
| Westmere | wes | 1.00 | 12 | 22.5 |
| Sandy Bridge | san | 1.82 | 16 | 30 |
| Ivy Bridge | ivy | 2.52 | 20 | 62 |
| Haswell | has | 3.34 | 24 | 122 |

## 5.4 Building Wind-US on the NAS

1. Download the Wind-US Build Distribution from IVMS.
   This distribution contains all of the necessary run scripts and source files needed to compile and run Wind-US.
   - Login to IVMS.
   - Select the Projects tab on the top of the page.
   - Select Open Wind Development to reveal the list of Wind releases.
   - Select the Wind-US version via the push-button in the right hand column.
   - Select the Downloads tab on the top of the page.
   - Go to the bottom of the Downloads page and download the Wind-US Build Distribution. It may take a few seconds before the Save File dialog appears. When it does, save the file as:

     ```
     windus.build.tar.gz
     ```

2. Transfer the build bundle to NAS (columbia or pleiades).
   - Put the *windus.build.tar.gz* file in the (new) directory *$HOME/WINDUS* where the source will be installed. You should now have:

     ```
     $HOME/WINDUS/windus.build.tar.gz
     ```

3. Unpack the build bundle on NAS.

   ```
   cd $HOME/WINDUS
   gunzip -c windus.build.tar.gz | tar xvf -
   ```

   This should extract everything to the new subdirectory:

   ```
   $HOME/WINDUS/wind-dev
   ```

4. Update the NAS login scripts.
   Users of csh and tcsh shells must edit the *$HOME/.cshrc* or *$HOME/.tcshrc* file respectively and make sure the following lines appear:

   ```
   module load comp-intel/2013.5.192
   module load mpi-sgi/mpt.2.11r13
   setenv CFDROOT   "$HOME/WINDUS/wind-dev"
   setenv WIND_DEV  "$HOME/WINDUS/wind-dev"
   source            $HOME/WINDUS/wind-dev/bin/cfd.login
   ```

   Similarly, *bash*, *ksh*, and *sh* users must edit their *$HOME/.profile* file and add make sure the following lines appear:

```
module load comp-intel/2013.5.192
module load mpi-sgi/mpt.2.11r13
CFDROOT   = "$HOME/WINDUS/wind-dev"           ; export CFDROOT
WIND_DEV  = "$HOME/WINDUS/wind-dev"           ; export WIND_DEV
.            $HOME/WINDUS/wind-dev/bin/cfd.profile
```

This sets up the Intel Fortran/C compilers, loads SGI's MPI message passing toolkit, and causes the contents of the *cfd.login* (or *cfd.profile*) file to be executed automatically for each new shell instance. The environment variables `CFDROOT`, `WIND_DEV`, `SYSTEM`, and `SYSTEM_CPU`, that are required for installing and running Wind-US will also be set, and the `PATH` will be modified to include the location of the Wind-US executable directories.

5. Test the NAS login scripts.
   At this point, log out and log back in. Check to see if the environment variables have been set to appropriate values, by doing:

   ```
   printenv CFDROOT
   printenv WIND_DEV
   printenv SYSTEM
   printenv SYSTEM_CPU
   ifort --version
   icc   --version
   which mpiexec
   ```

   They should have values similar to:

   ```
   $HOME/WINDUS/wind-dev
   $HOME/WINDUS/wind-dev
   LINUX64-GLIBC2.11
   XEON
   ifort (IFORT) 13.1.3 20130607
   icc (ICC) 13.1.3 20130607
   /nasa/sgi/mpt/2.11r13/bin/mpiexec
   ```

   (Note that `$HOME` may be expanded to your full home directory path.) If the variables are not set, or not set correctly, go back to step 4 and try again.

6. Move into the Wind-US build directory.

   ```
   cd $WIND_DEV
   ```

   This should put you in the *$HOME/WINDUS/wind-dev* directory.

7. Configure the makefiles.
   If you plan to build both Wind-US and the tools, then follow the instructions below for unpacking the tools distribution before making any changes to the makefiles. The reason for this is that the tools distribution also contains a copy of the makefiles and will overwrite any changes you might make here.

   The default values in the configuration files should be sufficient to produce an executable. To be safe, review the contents of the following files or parts of files, where *SYSTEM* and *SYSTEM_CPU* correspond to the `SYSTEM` and `SYSTEM_CPU` environment variables, paying special attention to the items noted below.

   - *Makefile.configure*
     - Set `USE_MPI=YES`.

  - Set `USE_PVM=YES`.
  - Set `BUILD_PVM=YES`.
  - The value of `PRECISION` may be changed to make all Fortran variables single or double precision. The default is a mix, with single precision for the mean flow solution and boundary conditions, and double precision for chemistry data, right-hand-side data, grid coordinates, and grid metrics.

- *source/Makefile.user*
  - Only examine this file if you experience problems during the build.
- *source/makefiles/Makefile.include.SYSTEM.SYSTEM_CPU.opt*
  - This file contains the compiler optimizations for the specific system being used.
  - If this makefile does not exist, you will need to create it. Usually the easiest way to do so is to copy and modify one of the existing files. For example:

```
cp -p source/makefiles/Makefile.include.LINUX64-GLIBC2.4.XEON.opt
      source/makefiles/Makefile.include.LINUX64-GLIBC2.11.XEON.opt
```

  - Check the definition of `CPP`, which is the full path name for the C pre-processor *cpp*, to make sure it is correct. You can locate *cpp* using:

```
whereis -b cpp
```

  - You may need to change the definitions related to the Fortran and C compilers being used (i.e., the variables `ABI`, `FC`, `F90`, `LD`, etc.). There is coding in the Linux makefiles for various Fortran and C compilers. These lines should be commented/uncommented as needed, but additional changes may also be necessary or desired. At the time of this writing, the following Intel 13.1.3 compiler settings were used:

```
ABI=       -Zp8 -pc80 -fp-model strict -fno-alias -heap-arrays \
           -fpic -traceback

FC=        ifort
FCOMP=     $(FC) $(ABI) -pad -ip -DINTEL
FFHOP=     -O3 -xSSE4.2 -axAVX
FFOPT=     -O3 -xSSE4.2 -axAVX
FFLOW=     -O1
FFNOP=     -O0

F90=       ifort $(ABI) -pad -ip -DINTEL
F90FHOP=   -O3 -xSSE4.2 -axAVX
F90FOPT=   -O3 -xSSE4.2 -axAVX
F90FLOW=   -O1
F90FNOP=   -O0

CC=        icc
CCOMP=     $(CC) $(ABI)
ANSI=      -ansi
POSIX=     -D_POSIX_SOURCE
CFOPT=     -O2 -xSSE4.2 -axAVX

CCP=       icpc $(ABI)
```

```
CPFOPT=     -O2 -xSSE4.2 -axAVX

LD=         ifort $(ABI) -O3 -ip -pad -xSSE4.2 -axAVX
```

The compiler flags `-xSSE4.2` and `-axAVX` are specific optimizations for the Westmere, Sandy Bridge, and Ivy Bridge processors that set the baseline code path and alternate optimized code paths respectively. For more information, visit:

* http://www.nas.nasa.gov/hecc/support/kb/Recommended-Compiler-Options_99.html

– If your preferred compiler doesn't support the Fortran long integer data type, add "`-DNOFLONG`" (without the quotes) to the definition of `WINDDEFS`. Errors in the compilation of *mem_management_module.f90* about an ambiguous definition for the generic interfaces `alloc` and `dealloc`, involving the specific interfaces for `allockindlong_1` and `allockindint_1`, etc., are an indication that "`-DNOFLONG`" is needed.

– If necessary, specify the location of the MPI libraries.

```
MPILIBS=        -L/nasa/sgi/mpt/2.11r13/lib -lmpi
```

The SGI MPI module makes available an `mpif90` command that automatically passes the library information to the Intel compiler through the `LD_LIBRARY_PATH`, `LIBRARY_PATH`, and `F_PATH` environment variables. However, all code compiled with this command will include the MPI libraries, even if they are not needed. This has been found to cause problems with some of the tools to be compiled below. Using `ifort` as the compiler directive and setting the `MPILIBS` makefile variable ensures that the MPI libraries will only be included in the Wind-US executable.

8. Compile the source code.

- Create a PBS script (*$WIND_DEV/make.wind.pbs*) to compile Wind-US. For *csh* and *tcsh* use the following:

```
#PBS -lselect=1:ncpus=4,walltime=2:00:00
cd $PBS_O_WORKDIR
echo "CFDROOT    = $CFDROOT"      |& tee    make.wind.log
echo "WIND_DEV   = $WIND_DEV"     |& tee -a make.wind.log
echo "SYSTEM     = $SYSTEM"       |& tee -a make.wind.log
echo "SYSTEM_CPU = $SYSTEM_CPU"   |& tee -a make.wind.log
cd $WIND_DEV
make opt |& tee -a make.wind.log
```

For *sh* and *ksh* use the above, but replace "`|&`" with "`2&>1 |`".

- Submit the job to compile the code.

```
cd $WIND_DEV
qsub -q devel make.wind.pbs
```

This may take roughly 30 minutes once your job begins running. You can use the `qstat` command to check on the status.

- If Wind-US compiled successfully you should now have:

```
$WIND_DEV/$SYSTEM/$SYSTEM_CPU/bin/Wind-USalpha.exe
```

If the executable was not produced, then examine the log file (*$WIND_DEV/make.wind.log*) for compilation errors.

9. Install the executables and scripts.
   If you have a previous installation of Wind-US that you wish to update (i.e., `CFDROOT` points to some directory other than the `WIND_DEV` you are building from), you will need to install the Wind-US and PVM executables. To do that, issue the commands

   ```
   make install
   make install_scripts
   make copy_pvm
   ```

   This will:

   - Copy the Wind-US executable to: `$CFDROOT/$SYSTEM/$SYSTEM_CPU/bin`
   - Copy the Wind-US run scripts to: `$CFDROOT/bin`
   - Copy the PVM executables and libraries to: `$CFDROOT/pvm/$SYSTEM/$SYSTEM_CPU`
   - Copy the PVM include files to: `$CFDROOT/pvm/include`
   - Copy the PVM scripts to: `$CFDROOT/pvm/lib`

10. If this is a new installation, it would probably be best to log out and log back in before running Wind-US. This executes the shell start-up scripts, modifying the `PATH` environment variable to include the newly-created location for the Wind-US executable.

11. Run the `wind` script.

    ```
    wind
    ```

    The new executable should appear in the list of available versions.

    ```
                    Select the desired version
    0: Exit wind
    1: Wind-US Alpha
    ```

    See Section 5.6 for running Wind-US on the NAS.

## 5.5 Building the Tools on the NAS

Unlike the Wind-US build distribution, in order to obtain the source for all the tools several downloads are required. The smaller tools are all bundled together and may be acquired from the "Downloads" page of the "Tools Makefiles" project. GMAN, CFPOST, and MADCAP are normally downloaded separately from their respective "Downloads" pages. Note that the Project Names for these are "gmanpre", "cfpost_prod", and "Madcap production", respectively. The instructions below only describe how to install the smaller utilities and CFPOST on NAS, since GMAN and MADCAP are more graphical in nature and typically not used over a remote connection.

Each build distribution is designed to be a completely independent package, so that the tools can be built without requiring any additional files from IVMS.[4] Thus, one could have separate directory trees for the Wind-US build distribution and each of the tools build distributions. This would lead to a great deal of duplication, however. Therefore, the build distributions are designed to overlay one another. The following instructions assume that all the tools are being built in the same tree.

[4]There are some exceptions to this, such as CFPOST, described below.

Note that the build procedure for the tools is somewhat more complicated than the Wind-US build process. Please report problems to the NPARC Alliance User Support Team at `nparc-support@arnold.af.mil` or (931) 454-7885.

1. Download the source files.
   The Tools Makefiles contains the source code for most of the smaller utilities.

   - Login to IVMS.
   - Select the Projects tab on the top of the page.
   - Select Open Wind Tools to reveal the list of tools.
   - Select Tools Makefiles via the push-button in the right hand column.
   - Select the Downloads tab on the top of the page.
   - Go to the bottom of the downloads page and download the Tools Makefiles Build Distribution. It may take a few seconds before the Save File dialog appears. When it does save the file as:

     `tools.build.tgz`

   The CFPOST source code must be downloaded separately.

   - Login to IVMS.
   - Select the Projects tab on the top of the page.
   - Select Open Wind Tools to reveal the list of tools.
   - Select cfpost_prod via the push-button in the right hand column.
   - Select the Downloads tab on the top of the page.
   - Go to the bottom of the downloads page and download the Tools Makefiles Build Distribution. It may take a few seconds before the Save File dialog appears. When it does save the file as:

     `cfpost_prod.build.tar.gz`

   Note that CFPOST requires the Madcap library files, which should be included in this cfpost build bundle.

   Some of the tools require lua and cgnslib header files and/or libraries.

   - Download lua (4.0.1) from `http://www.lua.org/ftp/lua-4.0.1.tar.gz` and save as:

     `lua.4.0.1.tar.gz`

     Due to changes in the API, newer versions of lua will not work with the Wind-US tools!!!
   - Download cgnslib (2.5.5) from `https://cgns.github.io/download.html` and save as:

     `cgnslib.2.5.5.tar.gz`

     Newer versions of cgnslib may also work.

2. Transfer the source code to NAS (columbia or pleiades).

   - Put the **.tar.gz* files in the (existing) directory *$HOME/WINDUS* where the source will be installed. You should now have:

```
$HOME/WINDUS/tools.build.tar.gz
$HOME/WINDUS/cfpost_prod.build.tar.gz
$HOME/WINDUS/lua.4.0.1.tar.gz
$HOME/WINDUS/cgnslib.2.5.5.tar.gz
```

3. Unpack the source code on NAS.
   Note that this will overwrite any changes you made to the Wind-US makefiles!

   ```
   cd $HOME/WINDUS
   gunzip -c tools.build.tar.gz | tar xvf -
   gunzip -c cfpost_prod.build.tar.gz | tar xvf -
   gunzip -c lua.4.0.1.tar.gz | tar xvf -
   gunzip -c cgnslib.2.5.5.tar.gz | tar xvf -
   ```

   This should extract everything to the following subdirectories:

   ```
   $HOME/WINDUS/wind-dev/tools-dev/*
   $HOME/WINDUS/wind-dev/tools-dev/cfpost_prod
   $HOME/WINDUS/lua-4.0.1
   $HOME/WINDUS/cgns_2.5
   ```

4. Update the NAS login scripts.
   Users of *csh* and *tcsh* shells must edit the *$HOME/.cshrc* or *$HOME/.tcshrc* file respectively and make sure the following lines appear:

   ```
   module load comp-intel/2013.5.192
   module load mpi-sgi/mpt.2.11r13
   setenv CFDROOT   "$HOME/WINDUS/wind-dev"
   setenv WIND_DEV  "$HOME/WINDUS/wind-dev"
   setenv TOOLSROOT "$HOME/WINDUS/wind-dev/tools-dev"
   source            $HOME/WINDUS/wind-dev/bin/cfd.login
   source            $HOME/WINDUS/wind-dev/tools-dev/bin/tools.login
   ```

   Similarly, *bash*, *ksh*, and *sh* users must edit their *$HOME/.profile* file and make sure the following lines appear:

   ```
   module load comp-intel/2013.5.192
   module load mpi-sgi/mpt.2.11r13
   CFDROOT   = "$HOME/WINDUS/wind-dev"           ; export CFDROOT
   WIND_DEV  = "$HOME/WINDUS/wind-dev"           ; export WIND_DEV
   TOOLSROOT = "$HOME/WINDUS/wind-dev/tools-dev" ; export TOOLSROOT
   .            $HOME/WINDUS/wind-dev/bin/cfd.profile
   .            $HOME/WINDUS/wind-dev/tools-dev/bin/tools.profile
   ```

   This causes the contents of the *tools.login* (or *tools.profile*) file to be executed automatically at login time to set up the environment variable `TOOLSROOT` that is required for installing and running the Wind-US tools, and to modify `PATH` to include the location of the proper executable.

5. Test the NAS login scripts.
   At this point, log out and log back in. Check to see if the environment variables have been set to appropriate values, by doing:

   ```
   printenv CFDROOT
   printenv WIND_DEV
   printenv TOOLSROOT
   ```

```
printenv SYSTEM
printenv SYSTEM_CPU
ifort --version
icc   --version
```

They should have values similar to:

```
$HOME/WINDUS/wind-dev
$HOME/WINDUS/wind-dev
$HOME/WINDUS/wind-dev/tools-dev
LINUX64-GLIBC2.11
XEON
ifort (IFORT) 13.1.3 20130607
icc (ICC) 13.1.3 20130607
```

(Note that `$HOME` may be expanded to your full home directory path.)

If the variables are not set, or not set correctly, go back to step 4 and try again.

6. Compile Lua.

```
cd $HOME/WINDUS/lua-4.0.1
make
```

This should create the following files:

```
$HOME/WINDUS/lua-4.0.1/bin/lua
$HOME/WINDUS/lua-4.0.1/bin/luac
$HOME/WINDUS/lua-4.0.1/lib/liblua.a
$HOME/WINDUS/lua-4.0.1/lib/liblualib.a
```

7. Compile the CGNS library.

```
cd $HOME/WINDUS/cgnslib_2.5
./configure --prefix=$HOME/WINDUS/cgnslib_2.5 --with-system=LINUX64 \
            --enable-64bit
make SYSTEM=LINUX64
mkdir include
mkdir lib
make install
```

This should create the following files:

```
$HOME/WINDUS/cgnslib_2.5/include/cgnslib.h
$HOME/WINDUS/cgnslib_2.5/include/cgnslib_f.h
$HOME/WINDUS/cgnslib_2.5/include/cgnswin_f.h
$HOME/WINDUS/cgnslib_2.5/lib/libcgns.a
```

8. Move into the Wind-US build directory.

```
cd $WIND_DEV
```

This should put you in the *$HOME/WINDUS/wind-dev* directory.

9. Configure the makefiles.

Review the contents of the following files or parts of files, where *SYSTEM* and *SYSTEM_CPU* correspond to the `SYSTEM` and `SYSTEM_CPU` environment variables, paying special attention to the items noted below.

- *Makefile.configure*
  - Make the modifications listed above in the instructions for building Wind-US on NAS.
- *source/makefiles/Makefile.include.SYSTEM.SYSTEM_CPU.opt*
  - Make the modifications listed above in the instructions for building Wind-US on NAS.
  - Comment out the explicit static and shared library settings:

    ```
    #TOOLS_STATLIB= -static
    #TOOLS_SHARLIB= -shared
    ```

    The default behavior of the Intel compiler is to use shared libraries.
  - Provide the proper locations of Tcl, Lua, and CGNS:

    ```
    TOOLS_TCLLIBS=  -L/usr/lib64 -ltcl8.5
    TOOLS_LUALIBS=  -L$(HOME)/WINDUS/lua-4.0.1/lib -llua -llualib
    TOOLS_CGNSLIBS= -L$(HOME)/WINDUS/cgnslib_2.5/lib -lcgns
    TCL_INCLUDE=    $(INCCMD)/usr/include
    LUA_INCLUDE=    $(INCCMD)$(HOME)/WINDUS/lua-4.0.1/include
    CGNS_INCLUDE=   $(INCCMD)$(HOME)/WINDUS/cgnslib_2.5/include
    ```
  - Provide the proper locations of the Intel libraries:

    ```
    TOOLS_OSLIBS=   -L/nasa/intel/Compiler/2013.5.192/lib/intel64 \
                    -lifcore -lirc -limf -lpthread
    TOOLS_CPPLIBS=  -L/nasa/intel/Compiler/2013.5.192/lib/intel64 \
                    -lifcore -lirc -limf
    MADCAP_OSLIBS=  -L/nasa/intel/Compiler/2013.5.192/lib/intel64 \
                    -lifcore -lcxa -lunwind -lpthread -lstdc++
    ```

    Depending on which version of the Intel compiler you choose, the library files may be in a different subdirectory below */nasa/intel*. These library paths might already be present in the enviroment variables.
- *source/makefiles/pvm_conf/SYSTEM.SYSTEM_CPU.def.opt*
- *tools-dev/Makefile*
- *tools-dev/Makedefs.SYSTEM*
- If GMAN is being built: *tools-dev/gmanpre/Makedefs.SYSTEM*
- If CFPOST is being built: *tools-dev/cfpost_prod/Makedefs.SYSTEM*
- If MADCAP is being built: *tools-dev/libmadcap/Makedefs.SYSTEM* and *tools-dev/madcapprod/Makedefs.SYSTEM*
- If GMAN, CFPOST, or MADCAP is being built: *tools-dev/libmdgl/SYSTEM.mkf*
  - If this makefile does not exist, you will need to create it. Usually the easiest way to do so is to copy and modify one of the existing files. For example:

    ```
    cp -p tools-dev/libmdgl/LINUX64-GLIBC2.7.mkf
          tools-dev/libmdgl/LINUX64-GLIBC2.11.mkf
    ```
  - Use the settings for the Intel compiler, if they are not already the default.

```
ABI=    -mcmodel=medium -pad -pc80 -fp-model strict -fno-alias
CC=     icc
CFLOPT=
```

Note that `CFLOPT` is defined to be empty.

10. Compile the tools source.

    On NAS, the front end node has the openmotif-devel-* package installed, which makes available a number of header files needed to compile the Wind-US tools. The worker nodes only have the library files installed. This means that the tools must be compiled on the front end node. So, instead of using a batch script like that to compile Wind-US, simply compile the tools from the command line.

    Make sure you are in the build directory.

    ```
    cd $WIND_DEV
    ```

    Next, *csh* and *tcsh* users should do

    ```
    make all_tools |& tee make_tools.log
    ```

    while *sh* and *ksh* users should do

    ```
    make all_tools 2>&1 | tee make_tools.log
    ```

    Tools can also be compiled individually, by doing

    ```
    make tool_name
    ```

    where *tool_name* is the name of the tool. Note that the names to be used for GMAN, CFPOST, and MADCAP are *gmanpre*, *cfpost_prod*, and *madcapprod*, respectively.

    After compilation is complete, the following programs should appear in directory *$WIND_DEV/$SYSTEM/$SYSTEM_CPU/bin*

    ```
    USintrpltQ.exe   cfreset_iter.exe  gpro.exe     rplt3d
    adfedit          cfrevert.exe      gridvel.exe  scan
    cfappend.exe     cfsequence.exe    icees        terp
    cfaverage.exe    cfspart           jormak.exe   thplt.exe
    cfbeta.exe       cfsplit.exe       lstvars      timplt.exe
    cfcnvt           cfsubset.exe      mpigetnzone  tmptrn.exe
    cfcombine.exe    cfunsequence.exe  newtmp       usplit-hybrid.exe
    cflistnum        cfview.exe        npair        windpar.exe
    cfnav.exe        chmgr.exe         parcnl       wnparc
    cfpart.exe       decompose.exe     readcf       wplt3d
    cfpost_prod.exe  delvars           resplt.exe   writcf
    cfreorder.exe    fpro              rnparc
    ```

    Depending on the size of the array parameters requested, the `thplt.exe` executable might not get created with the default memory model. If it was not created, then edit the file *$WIND_DEV/source/makefiles/Makefile.include.$SYSTEM.$SYSTEM_CPU.opt* to use the following ABI settings:

    ```
    ABI=       -Zp8 -pc80 -fp-model strict -fno-alias -heap-arrays \
               -mcmodel medium -traceback
    ```

    and recompile just that tool. From the build directory, remove the old object file.

```
cd $WIND_DEV
rm -f OBJECTS/$SYSTEM/$SYSTEM_CPU/thplt.o
```

Next, *csh* and *tcsh* users should do

```
make thplt |& tee make_thplt.log
```

while *sh* and *ksh* users should do

```
make thplt 2>&1 | tee make_thplt.log
```

Check *$WIND_DEV/$SYSTEM/$SYSTEM_CPU/bin* to confirm that `thplt.exe` was created.

11. In order for the tool scripts to locate the executables, they must be installed in the proper location. To install the executables, do:

    ```
    make install_tools
    ```

    This copies the tools executables to *$TOOLSROOT/$SYSTEM/$SYSTEM_CPU/bin*.

12. If this is a new installation, it would probably be best to log out and log back in again before running any of the tools. This executes the shell start-up scripts, modifying the `PATH` environment variable to include the newly-created location for the tools executables.

## 5.6 Running Wind-US on the NAS

1. Make a directory containing your Wind-US input files:

   ```
   run.dat
   run.cgd
   run.mpc
   run.lis  (if continuing from a previous solution)
   run.cfl  (if continuing from a previous solution)
   ```

   The *run.mpc* file should have the following form:

   ```
   / Wind-US parallel processing file for NAS.
   / Currently set to use 2 nodes with 12 processors each
   /                  and 1 node  with  6 processors
   /                  for a total of   30 cores.
   /
   host localhost nproc 12
   host localhost nproc 12
   host localhost nproc 6
   ```

   Each different type of NAS compute node has a different number of processors. For example, the Westmere nodes have 12 processor cores. Therefore, each host entry above has at most `nproc 12`. The user will need to experiment to determine whether the best performance is obtained when all of the processor cores on a given host are used (12+12+6=30) or when the hosts are most closely balanced (10+10+10=30). The difference between internode and intranode communication might be mesh dependent.

   Users might also want to include a `checkpoint` command in the above file so that the worker solutions are sent to the master process at regular intervals. Please see the Wind-US User's Manual for more details on the format and features of the parallel processing file.

2. Start the Wind-US script with one of the following commands:

```
wind -runinplace -cl -usessh -mpmode MPI -mpiver SGI
wind -runinplace -cl -usessh -mpmode PVM
```

- To specify a non-default NAS charge number, add the `-grpcharge` option and charge code to the Wind-US command line.
- Follow the prompts for your input, output, mesh, and flow files or specify them on the command line using the proper syntax.
- When asked, indicate that you want to run in multi-processor mode.
- If prompted, enter the number of zones in the cgd file.
- When prompted for the type of queue, choose QSUB_PBS_QUE.
- Enter the queue name: i.e., `long` for the long queue.
- Enter the solver run wall clock time.
- Enter the termination processing time.
- Enter the number of nodes to run on. This should match the number of `host` entries in your mpc file.
- Enter the number of processors per node to use.
- Enter the number of MPI processes per node to use. This is typically the same as the number of processors per node. For the default setting of `ASSIGNMENT MODE DEDICATED` in the *.mpc* file, each zone should have its own MPI process. One additional MPI process is required for the master process. You must remember to account for this extra process when answering the prompts or your run will fail to initialize MPI.
- Enter any optional attributes. For example, to specify that the job should run on Westmere nodes use the following:

  ```
  model=wes
  ```

  The model names for other processor types are listed in Table 1.
- When prompted to "Press CR to submit job, or another key (except space) and CR to abort," do the latter. This will create a file called *run.job.pl*.

The maximum solver execution time is determined by subtracting the termination processing time from the solver run wall clock time. When the Wind-US run job is submitted, it will create a *preNDSTOP* file. When the maximum solver execution time has expired, this file will be renamed *NDSTOP*, forcing Wind-US to begin a graceful shutdown.

Users should make sure that the termination processing time is sufficient to allow Wind-US to complete the termination process. At the end of the **.lis*, there is a summary indicating the time spent during execuation and termination.

Users should also make sure that they request adequate time from the queue in which they submit their jobs. This is detailed in the next step. Otherwise, the queue will terminate Wind-US, resulting in a less than graceful shutdown.

3. Edit *run.job.pl*. If you see a line like the following near the top of the file:

   ```
   #PBS -l nodes=1234:ppn=2
   ```

   replace it with

```
#PBS -l select=2:ncpus=12+1:ncpus=6
#PBS -l walltime=40:00:00
#PBS -m e
```

This will select 2 nodes with 12 cpus and 1 node with 6 cpus, which matches the request in the *run.mpc* file. The walltime can be adjusted as desired (hh:mm:ss) or as an integer number of seconds, and the last line will send you an email when your job is completed.

Make sure to request at least as much walltime as was specified in the Wind-US prompts above, because the queuing system does not terminate jobs as cleanly as Wind-US does.

4. If you plan on resubmitting the same job again later (i.e., you want to run 10000 cycles now and 10000 cycles later) you can save a copy of the run script.

   ```
   cp -p run.job.pl run.job.pl.bak
   ```

   To resubmit later, you can skip the above steps and simply reuse the run script.

   ```
   cp -p run.job.pl.bak run.job.pl
   ```

   Note that if you increase the number of cycles in your **.dat* file, you may need to adjust the run time specified in the job file.

5. Submit the job to the `long` queue with the command:

   ```
   qsub -q long run.job.pl
   ```

   Some other useful commands are:

| Command | Action |
|---|---|
| `qstat -q` | List all queue names and run limits. |
| `qstat -a long` | List all jobs running in the `long` queue. |
| `qstat -u USER` | List all jobs running for username `USER`. |
| `qdel JOBNAME` | Delete job with name `JOBNAME`. Useful if Wind-US has not yet started. |

6. In order to improve I/O performance for large jobs, Wind-US 3.0 uses a newer ADF library than its predecessors. Grid and solution files used with Wind-US 3.0 will automatically be upgraded to the new format, and the tools compiled in the above steps will also work with the new file structure. However, if you transfer the grid or solution file(s) back to your local workstation your existing tools may not be able to read them. If you experience this problem, you should upgrade the tools at your local site.

# 6 Porting Wind-US to a New UNIX Platform[5]

## 6.1 Porting Guidelines

The source files for Wind-US (and the non-system libraries it depends on) are provided in the build distribution. See Section 2.2.2 for instructions on how to obtain copy. Put the gzip'ed tar file containing the build distribution into an appropriate directory. The same one used for the application and tools distributions would be a good choice. In that directory, unpack the file by doing:

```
gunzip -c filename | tar xvf -
```

where *filename* is the file name, including the *.tar.gz* extension. Once you have installed the source on your system, change directory to the *wind-dev* directory and set the `WIND_DEV` environment variable to point to that directory. For example, if the previous step was done in */usr/local/wind*, *csh* and *tcsh* users would do

```
cd wind-dev
setenv WIND_DEV /usr/local/wind/wind-dev
```

Now you are ready to begin the task of actually porting Wind-US. There are three files you need to either examine or, if they don't already exist, create:

- *$WIND_DEV/Makefile.Configure*

  This file contains generic information such as the name of the *make* utility, where to find the source for the various components of Wind-US, and whether or not to build PVM. In addition, this file defines the machine and CPU type for which Wind-US will be built. By default, these are set from the `SYSTEM` and `SYSTEM_CPU` environment variables. If this file doesn't exist, download the build distribution again — something went very wrong.

- *$WIND_DEV/source/makefiles/Makefile.include.$SYSTEM.$SYSTEM_CPU.opt*

  This file contains system-specific information, such as compiler names, optimization switches, machine-specific compilations, the name of the *awk* command, the location of the C preprocessor *cpp*, and extra libraries which must be linked in. Pay particular attention to the `MCHNSRCS` lines in the "Common File directory special rules" section. If you are compiling for a completely new machine, you may have to create one or more of these files (which are found in *$WIND_DEV/libcfd/machine.lib*). You may, however, be able to use files created for other machines.

  *Note*: There are other possible extensions than *.opt* for this file. If you wish to compile for use with a debugger, create a *Makefile.include...* with a *.dbx* extension. See the SGI files for examples. Another possible extension is *.pure*, which defines the compilation configuration for use with the *Purify* debugging software package.

- *$WIND_DEV/source/makefiles/pvm_conf/$SYSTEM.$SYSTEM_CPU.def.opt*

  This is the PVM configuration file. If your system is not currently supported, check in *$WIND_DEV/pvm/conf* to see if a configuration file for your system already exists. Note that the definition of `SYSTEM` may be different for PVM than for Wind-US. If you cannot find a pre-existing configuration file for your system, you will have to create one. Choose a file for a system similar to yours and use that as a starting point.

  *Note*: As before, there are other extensions besides *.opt*. See the SGI files for examples.

[5] The material in this section was originally written by Chris Nelson of Sverdrup Technology, Inc. - AEDC Group.

When all three files exist and everything appears in order, then simply type `make opt` (if you want to compile for optimization) and the make system should take care of the rest. If you type `make` by itself (or `make help`), a list of all the compilation options will be printed.

## 6.2 Frequently (or not) Encountered Problems

1. *The `SYSTEM` and `SYSTEM_CPU` variables are not being set in a way that makes sense for my system. What can I do?*

   It is likely that the *pvmgetarch* and *pvmgetcpu* scripts need to be modified to detect your particular system. These scripts are found in the *$CFDROOT/bin* directory. You will have to determine for yourself how the scripts can correctly distinguish your system from others, but the examples already in them should get you started.

   Once you have modified the files, you will need to copy them to several different places. Copy *$CFDROOT/bin/pvmgetarch* to *$CFDROOT/pvm/pvmgetarch-cfd* and also to *$WIND_DEV/pvm/lib/pvmgetarch-cfd*. Copy *$CFDROOT/bin/pvmgetcpu* to *$CFDROOT/pvm/pvmgetcpu-cfd*.

   Ideally these files should be links, and perhaps some day they will be.

2. *Can I cross-compile Wind-US for a different system using these makefiles?*

   If you have a compiler that is capable of compiling for many different machines/CPUs, then you may want to use a single machine to create executables for all of them. To some extent, this is possible, but it will take some extra work on your part.

   First, modify *$WIND_DEV/Makefile.configure* so that `SYSTEM_SUFFIX` and `SYSTEM_BLD_CPU` are set to the machine you wish to compile for. The extra work comes because the PVM compilation system is not set up for cross-compiling. Therefore, you must set `BUILD_PVM` to `NO` and obtain (by one means or another) PVM libraries and executables for each machine you wish to compile for. For SGI workstations with MIPS processors, the default is to compile PVM for the lowest common denominator CPU, so, for a given operating system, you should be able to use the same PVM files for R10000 (and up) machines.

3. *I only have a single processor machine. Do I really have to mess with all this parallel stuff?*

   No, you don't. To turn off the parallel capabilities of Wind-US, edit *$WIND_DEV/Makefile.configure* and set `BUILD_PVM` to `NO`. Next, edit *$WIND_DEV/source/Makefile.user* and remove `pssubs` from the `LINK_MODULES` line and add it to the `DUMMY_MODULES` line.

4. *The compilation gets all the way to the end, and then fails with complaints about `rcutv1`, `rcutaa`, and `rcimsc` being unresolved symbols. What gives?*

   This problem pops up on Sun workstations (and maybe others) because *awk* is not working as expected. Check to see if *nawk* is available on your system. If it is, edit *$WIND_DEV/source/makefiles/Makefile.include.$SYSTEM.$SYSTEM_CPU.opt* and set the `AWK` variable to `nawk`.

5. *I modified $WIND_DEV/source/makefiles/Makefile.include... (or $WIND_DEV/source/makefiles/pvm_conf/$SYSTEM...def... ), but when I re-compile, none of my changes are picked up. What is wrong?*

   The problem is that the files that are actually used for the compilation are not the ones under *$WIND_DEV/source/makefiles*. The actual files are:

*$WIND_DEV/Makefile.include.$SYSTEM.$SYSTEM_CPU* and *$WIND_DEV/pvm/conf/$PVMSYS.def*, where *$PVMSYS* is set by *$WIND_DEV/pvm/lib/pvmgetarch*. When you make changes, you must either copy the files to their final destination or "select" the makefiles for the type of build you're doing (using, for example `make select_opt` if you want to compile with optimization). When you run one of the "global re-build" compilations (e.g., `make opt`) then the "select" is done automatically. Ideally, the system should automatically check to see if any of the configuration files have been modified, but right now they don't.

6. *I modified $WIND_DEV/Makefile.include.SYSTEM.SYSTEM_CPU (or $WIND_DEV/pvm/conf/$PVMSYS.def), but when I tried to build Wind-US, it didn't seem to find my changes. I checked the files, and my changes were gone. What happened?*

   See the answer to #5, above. The short answer is that your changes were overwritten. You have to modify the files under *$WIND_DEV/makefiles* and "select" them in order to be sure that the changes will "stick".

7. *My make utility complains that there are errors and aborts before anything gets compiled. Why?*

   IBMs seem to be particularly bad about this. The "errors" are usually not errors in the sense that anything is catastrophically wrong, but rather, in the process of house-cleaning, the make system may be trying to remove files that don't exist or is checking to see if a file does exist when it doesn't. The solution is to modify *$WIND_DEV/Makefile.configure* and add a `-i` switch to the `MAKE` and `PVMMAKE` variables. Sometimes, switching to *gmake* will solve the problem, but be aware that the PVM make system has *make* hardwired. It may also be necessary to start the make process with the `-i` switch (e.g. `make -i opt`).

8. *When I try to compile, it gets to a certain point and then just hangs. What is the problem?*

   The system of makefiles used to build Wind-US is pretty complex, and some versions of *make* just can't handle this complexity. The weakest link appears to be in the PVM build. If your *make* utility is not up to the task, try using another one (such as *gmake*). Since the PVM makefiles are hardwired to use *make*, you may have to use an alias rather than changing *Makefile.configure*. For example, on the HP/Convex CSPP system, it might be necessary to alias *make* to *gmake*.

9. *I'm getting undefined symbols at the end of my compilation, but it's more than just the three you mentioned in question #4. What are likely causes of the problem?*

   The most likely explanation is that your system does not load all the libraries you need by default. Run *grep* on some of the symbols that it complains about (ignoring post-pended underscores — i.e. `psexit`, not `psexit_`) and see if the routines are from within Wind-US:

   ```
   cd $WIND_DEV/source
   grep the_unknown_symbol */*
   ```

   If that fails to find anything, check in *libcfd*:

   ```
   cd $WIND_DEV/libcfd
   grep the_unknown_symbol */*
   ```

   If you still can't find it, try *libadf*:

   ```
   cd $WIND_DEV/libadf
   grep the_unknown_symbol *
   ```

If that fails as well, then hunt through the *$WIND_DEV/pvm* subdirectories in a similar fashion. Once you are satisfied that the symbol is not from anything in the Wind-US distribution, you will have to poke around the system libraries to identify which ones should be added to the `EXTRALIBS` line in *$WIND_DEV/source/makefiles/Makefile.include.$SYSTEM.$SYSTEM_CPU.opt* (or *.dbx* etc.).

If the symbols are from within Wind-US, you need to look back through the compiler listing to see what messages were output when the library which contains that routine was compiled to see what went wrong.

10. *I'm having trouble compiling the ADF library. What can I do?*

    The ADF core library (*libadf*) has only been ported to a finite number of machines. If your machine is not one of them, you may have to add some lines to *ADF_fbind.h* for your system.

11. *I'm getting some fairly bizarre compiler errors when compiling the Common File library. What is going on?*

    As with the ADF library (see #10), the Common File library has only been ported to a limited number of machines. Check in *$WIND_DEV/libcfd/include/bind_f_and_c.h* to see if definitions for your system are there. If not, you will have to add them.

12. *Okay, I got libadf and libcfd to compile, but now Wind-US itself is complaining. Where should I look?*

    As with *libadf* (see #10) and *libcfd* (see #11), you may need to add lines appropriate for your system to a header file. In this case, it's *$WIND_DEV/source/include/fbind.h*.

13. *I'm having trouble getting PVM to compile and run properly. What should I do?*

    If the problem seems to be with the *make* system itself, then you may be able to successfully compile "manually" by using the following procedure (modify as needed for your particular shell):

    ```
    cd $WIND_DEV/pvm
    setenv PVM_ROOT $cwd
    make clean
    make
    ```

    If that works, then copy the PVM libraries (in *$WIND_DEV/pvm/lib/$PVMSYS*) to the `$LIBDIR` defined in *Makefile.configure*. Also remember to copy the PVM executables to *$CFDROOT/pvm/$SYSTEM/$SYSTEM_CPU*.

    An additional possibility is that you have another version of PVM already installed, and certain environment variables may be set for that version which conflict with the version of PVM shipped with Wind-US. The solution is to make sure that neither of the environment variables `PVM_ROOT` nor `PVM_ARCH` are set prior to compiling or running Wind-US. (The various scripts should set these variables as needed).

    If you still can't compile or run PVM, then you may need to talk to the PVM developers (see http://www.epm.ornl.gov/pvm/) to see about porting it to your system. In the meantime, you can still run Wind-US in single processor mode (see #3).

14. *I've gone through the whole process you describe above (in Section 6.1), but when I type* `make opt`*, I get one or more errors dealing with file permissions, like*

```
cp: cannot create /usr/local/wind/wind-dev/include/ADF.h:
Permission denied.
```

*What is the problem?*

By default (for security reasons, I believe), the source files only have "read" permission enabled. Thus, when the make system tries to do an operation which results in a "write" (which happens mostly when it's setting up the *include* directory), an error results and the system screeches to a halt. The solution is to make sure that you have write permission for all files and directories under *$WIND_DEV*.

15. *When running the Wind-US code on my brand new SGI R12000 system, I get the following error message*

```
 Program aborting due failure in common I/O library call.
 Subroutine called: CFRWFC
ADF 54: A node-id of 0.0 is not valid.
```

*What should I do?*

The cause of this problem has been traced to a change in the default floating point exception mode for R12000 systems. R10000 and R4400 SGI systems are not affected.

To run Wind-US on R12000 systems, you can upgrade to IRIX 6.5.4, and make sure that the kernel parameter "`fpcsr_fs_bit`" is equal to zero. After upgrading to IRIX 6.5.4, the value of this parameter may be determined by doing

```
systune fpcsr_fs_bit
```

If the value is non-zero, it should be changed by doing (as root)

```
systune fpcsr_fs_bit 0
```

The change in the value of `fpcsr_fs_bit` occurs dynamically, and does not require rebooting the system.

16. *I finally got everything to compile and link, but when I try to run the code, I get a "Program aborting due failure in common I/O library call" message, and then the code exits. What should I do?*

If you're running on an SGI R12000 system, see the previous question. Otherwise ...

Ironically, the weakest link in the Wind-US chain (as far as porting goes) has nothing to do with the solver algorithm, parallelization, or memory management (in the fluid solver). The weakest link is, in fact, the file I/O system. The problems all seem to center around the ADF core library. Even if you did not see a specific ADF error code (e.g., "ADF 54: A node-id of 0.0 is not valid"), if you can run the same case (with the same files) on another machine, the problem is almost certainly in *libadf*.

If this happens, please notify the code developers so that we can work on finding a fix. Obviously, if we don't have access to a machine of the type you are working on, then our ability to solve the problem is limited, but at least we can note the problem. If you feel energetic, you could also notify the CGNS folks over at https://cgns.github.io/ that you found a problem.

You can, of course, attempt to debug it yourself. If you do find a solution, please send it to us so that we can make the fix available to everyone. If, however, you don't have the time or the inclination for that, then your best bet is to convert your input Common Files (which are version 3.0 by default) to version 2.0. Version 2.0 of the Common File system does not use the

ADF core, and, so far, it has always worked when version 3.0 wouldn't. The way to convert the files is to use the *cfcnvt* utility. Choose option 3, "Compress a Common File", answer the questions, and then enter "2" when asked "Output CF version number (2 or 3 (default))".[6]

17. *When compiling the Common File library, I'm getting messages about an undefined function called "tempnam". What should I do?*

    Edit all *Makefile.include.$SYSTEM.$SYSTEM_CPU.** files (found in *$WIND_DEV/makefiles*). Look for a line that defines "CFDEFS". On this line add "-DNO_TEMPNAM". Next, "select" the appropriate compilation type (e.g., `make select_opt`) and re-compile.

[6] Note that this assumes you have access to a system for which the tools executables, including *cfcnvt*, are available.