WIND 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 December 5, 2007. Please send corrections, additions, ideas, etc., to Charlie Towne at *towne@grc.nasa.gov*.

Contents

1 Overview

The NPARC Alliance flow simulation system is distributed to users as a collection of compressed tar files, containing the WIND code and several pre- and post-processing tools. Both the WIND code 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 Application Distribution

The WIND *application* distribution includes all the executable programs and shell scripts necessary to run the WIND code on any supported computational platform. Pre-compiled WIND 5.0 executables are available for the following systems:

- Silicon Graphics
  - IRIX 6.5, R5000, R10000, and R12000 processors
  - Multi-processing IRIX 6.5, R10000, R12000, and R14000 processors
- Hewlett-Packard
  - HPPAMP11, PA-8500 processor
- Sun
  - Single and multi-processing SunOS 5.*x*, SPARC4U processor
- Single and multi-processing Linux, *x*86 processor

The WIND and PVM executables were created assuming *rsh* (*remsh* for HP systems) will be used as the “remote shell” command for parallel operation. If WIND is to be run in parallel mode on systems using some other remote shell command, like *ssh*, you'll need to build the WIND and PVM executables using the *build* distribution.

The WIND executable was also built without MPI message passing. If you're going to be running on a multi-processor system and using MPI, you'll need to build the WIND executable using the build distribution.

Tools Application Distribution

A tools *application* distribution includes all the executable programs and shell scripts necessary to run one or more of the various pre- and post-processing tools on any supported computational platform. This includes GMAN, a pre-processor used for setting boundary conditions and multi-zone connectivity, and CFPOST, a post-processor used to list and plot results, generate reports, and produce files for other post-processors. (See the *GMAN User's Guide* and the *CFPOST User's Guide*.) It also includes a variety of smaller utilities, described in the *WIND Utilities Guide*.

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

WIND Build Distribution

The WIND *build* distribution contains all the files needed to build and install the WIND code on any supported platform. This includes:

- source code for the WIND flow solver

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

- source code for all the required library routines, including PVM
- shell scripts needed to run WIND
- 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 any supported platform. This includes:

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

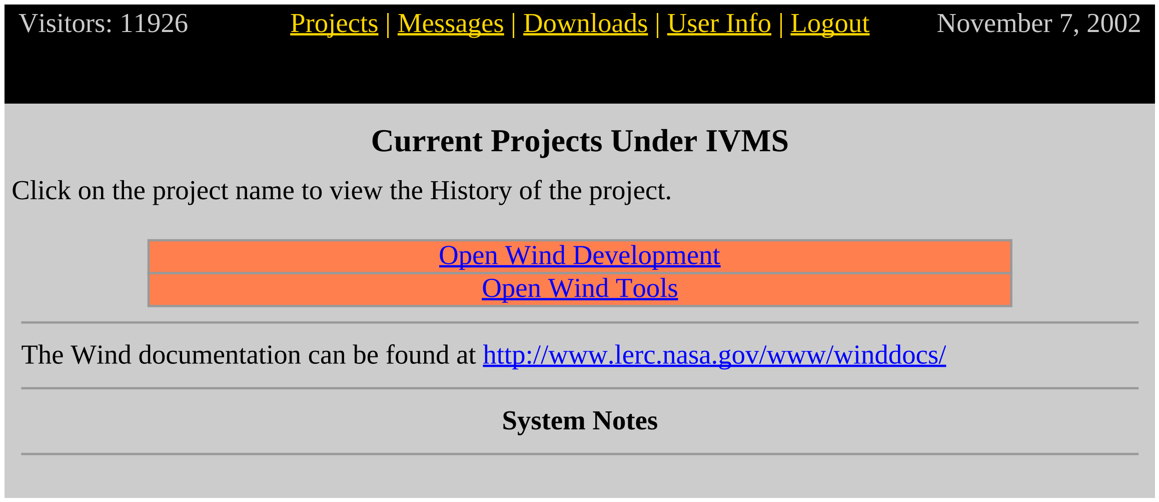
2 Obtaining the WIND Code and Tools

The WIND code, 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://web.arnold.af.mil/nparc`. You may also contact the NPARC Alliance User Support Team via email to `nparc-support@info.arnold.af.mil`, or by phone at (931) 454-7455.

2.1 IVMS (Internet Version Management System)

Approved users download the WIND code 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 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.


Visitors: 11926 Projects | Messages | Downloads | User Info | Logout November 7, 2002

Current Projects Under IVMS

Click on the project name to view the History of the project.

Open Wind Development

Open Wind Tools

The Wind documentation can be found at http://www.lerc.nasa.gov/www/winddocs/

System Notes

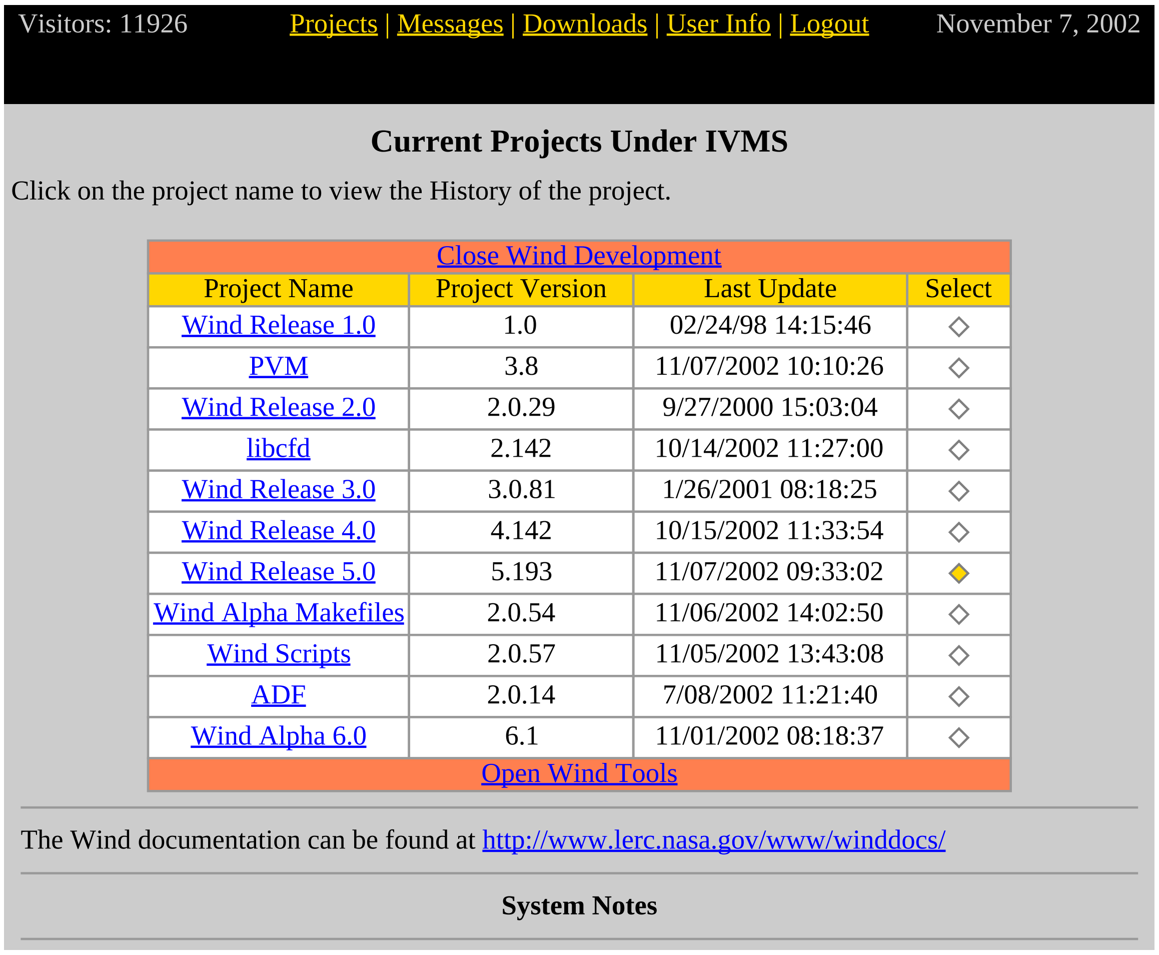
Figure 1: IVMS Projects Page.

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 the WIND code may be listed on the WIND Projects Page, shown in Figure 2.

Officially-released versions of WIND are listed as "Wind 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.



Visitors: 11926 Projects | Messages | Downloads | User Info | Logout November 7, 2002

Current Projects Under IVMS

Click on the project name to view the History of the project.

Close Wind Development

Project Name	Project Version	Last Update	Select
Wind Release 1.0	1.0	02/24/98 14:15:46	◇
PVM	3.8	11/07/2002 10:10:26	◇
Wind Release 2.0	2.0.29	9/27/2000 15:03:04	◇
libcfd	2.142	10/14/2002 11:27:00	◇
Wind Release 3.0	3.0.81	1/26/2001 08:18:25	◇
Wind Release 4.0	4.142	10/15/2002 11:33:54	◇
Wind Release 5.0	5.193	11/07/2002 09:33:02	◆
Wind Alpha Makefiles	2.0.54	11/06/2002 14:02:50	◇
Wind Scripts	2.0.57	11/05/2002 13:43:08	◇
ADF	2.0.14	7/08/2002 11:21:40	◇
Wind Alpha 6.0	6.1	11/01/2002 08:18:37	◇

Open Wind Tools

The Wind documentation can be found at http://www.lerc.nasa.gov/www/winddocs/

System Notes

Figure 2: WIND Projects Page.

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.

Beta and alpha versions are generally distributed only within the NPARC Alliance. They may be made available to others on a case-by-case basis, however, if a user or developer has a need for a specific feature not in the production version, or is actively participating in development of the Alliance-supported code. 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 code user is "WIND Release 5.0," containing the *application* and *build* distributions for the current production version of WIND. (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 WIND executable in the application distribution may be slightly out of date.

The WIND 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 Release 5.0	5.195	12/10/2002 11:16:54
HPPAMP11/PA8500	5.193	11/07/2002 09:32:57
LINUX/X86	5.193	11/07/2002 09:32:57
LINUXMP/X86	5.193	11/07/2002 09:32:58
SGI6.5/R5000	5.190	10/22/2002 13:22:06
SGI6.5/R10000	5.193	11/07/2002 09:32:59
SGI6.5/R12000	5.193	11/07/2002 09:32:59
SGIMP6.5/R10000	5.193	11/07/2002 09:32:59
SGIMP6.5/R12000	5.193	11/07/2002 09:32:59
SUNMP5/SPARC4U	5.193	11/07/2002 09:32:59
SUN5/SPARC4U	5.193	11/07/2002 09:33:00

From the table, the current version of the source code for WIND Release 5.0 is 5.195. The "Last Update" date shown for Wind Release 5.0 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 executable for SGI R10000 systems running IRIX 6.5 (for example) was created from WIND version 5.193, on November 7, 2002. Note that the latest compiled version isn't normally the same for all platform/CPU types.

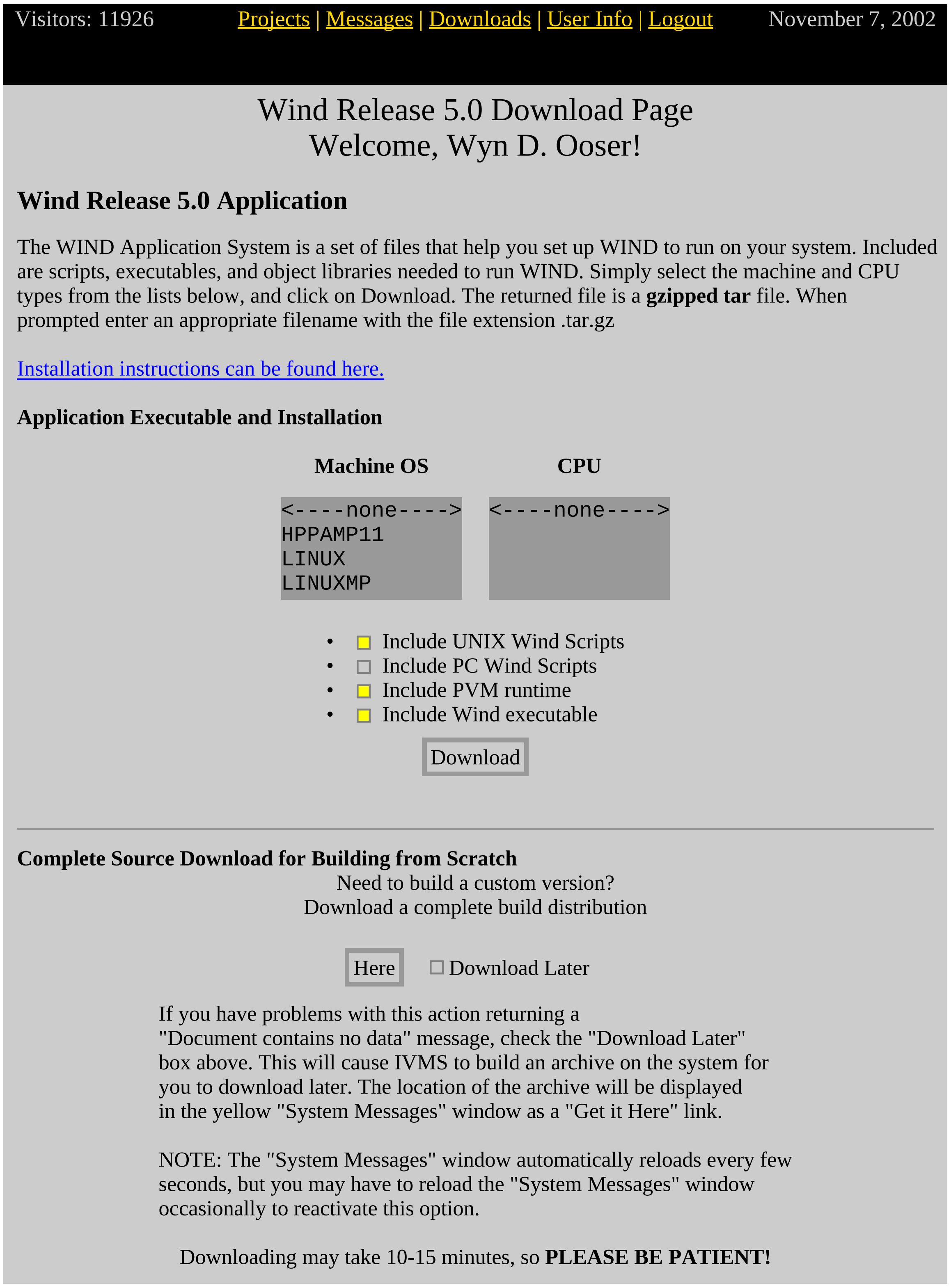
The changes made to the code since its release are described in the "Source History" page, following the above table. By examining the information in the "Source History" page, you can determine whether or not the currently available executable is suitable for your particular problem. If a change has 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 Application Distribution

To download the WIND application distribution, select "WIND Release 5.0" on the WIND Projects Page (see Figure 2) by clicking on the button in the right-hand column (note, however, that it should be already selected by default), then click on "Downloads" in the menu at the top of the page. You'll be connected to the WIND Download Page, like the one shown in Figure 3 for Dr. Wyn D. Ooser. The top part of this page deals with downloading the application distribution, i.e., the scripts and pre-compiled WIND 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.

Next, select (or de-select) the appropriate entries from the list located just above the "Download" button. For a UNIX system, new users should select the WIND executable, the PVM run-time scripts, and the UNIX WIND scripts. Note that the UNIX scripts are not platform-dependent, so


Visitors: 11926 Projects | Messages | Downloads | User Info | Logout November 7, 2002

Wind Release 5.0 Download Page
Welcome, Wyn D. Ooser!

Wind Release 5.0 Application

The WIND Application System is a set of files that help you set up WIND to run on your system. Included are scripts, executables, and object libraries needed to run WIND. Simply select the machine and CPU types from the lists below, and click on Download. The returned file is a **gzipped tar** file. When prompted enter an appropriate filename with the file extension .tar.gz

Installation instructions can be found here.

Application Executable and Installation

Machine OS	CPU
<----none---->	<----none---->
HPPAMP11	
LINUX	
LINUXMP	

- ☐ Include UNIX Wind Scripts
- ☐ Include PC Wind Scripts
- ☐ Include PVM runtime
- ☐ Include Wind executable

Download

Complete Source Download for Building from Scratch

Need to build a custom version?
Download a complete build distribution

Here ☐ Download Later

If you have problems with this action returning a "Document contains no data" message, check the "Download Later" box above. This will cause IVMS to build an archive on the system for you to download later. The location of the archive will be displayed in the yellow "System Messages" window as a "Get it Here" link.

NOTE: The "System Messages" window automatically reloads every few seconds, but you may have to reload the "System Messages" window occasionally to reactivate this option.

Downloading may take 10-15 minutes, so **PLEASE BE PATIENT!**

Figure 3: WIND Download Page.

when downloading executables for additional UNIX platforms and/or CPU types, only the WIND 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.

2.2.2 WIND Build Distribution

If a pre-compiled WIND 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 build distribution.

To download the WIND build distribution, simply click on the "Here" button in the bottom part of the WIND Download Page (see Figure 3). IVMS will package the necessary files into a gzip'ed tar file, and, if the "Download Later" button is not selected, you'll be prompted for the file name, which should end with the extension The whole process may take several minutes, depending on your Internet connection and the load on the IVMS server, so please be patient.

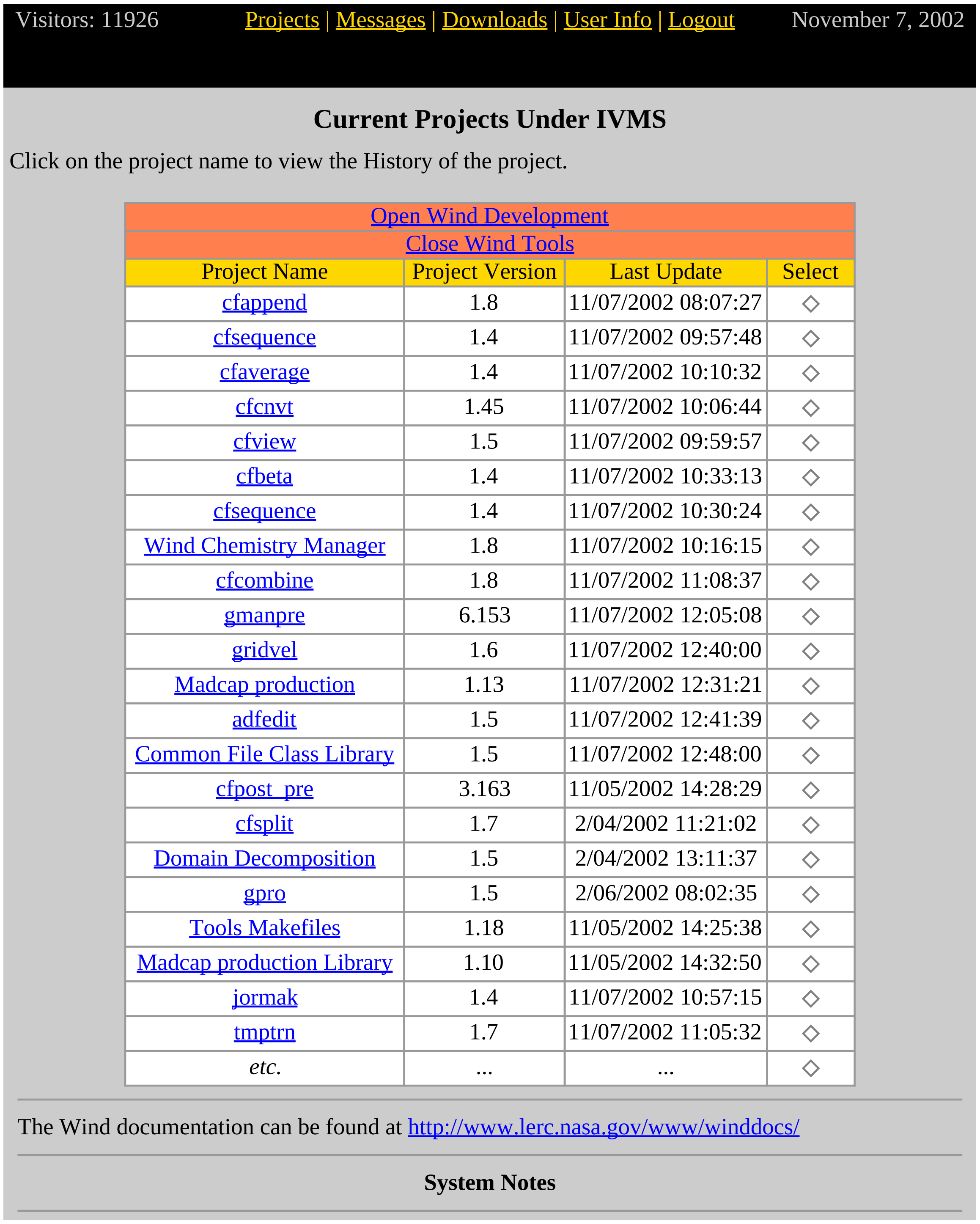
Under some conditions, this process may time out before completion, resulting in a "Document contains no data" error. The "Download Later" option was added to solve this problem. If the "Download Later" button is selected, IVMS will package the necessary files into a gzip'ed tar file, and a "Get it here" 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, 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 (partially) in Figure 4. Tools may be downloaded as a group, either as executables in a "tools application distribution" or as source code in a "tools build distribution". They may also be downloaded individually, either as an executable or source code.

At least one of these tools, GMAN (see the *GMAN User's Guide*) must be downloaded, since it's required for setting boundary condition types for input to WIND. Another, CFPOST (see the *CFPOST User's Guide*), is very useful in examining the results from WIND runs. Other tools are described in the *WIND Utilities Guide*.



Visitors: 11926 Projects | Messages | Downloads | User Info | Logout November 7, 2002

Current Projects Under IVMS

Click on the project name to view the History of the project.

Open Wind Development

Close Wind Tools

Project Name	Project Version	Last Update	Select
cfappend	1.8	11/07/2002 08:07:27	◇
cfsequence	1.4	11/07/2002 09:57:48	◇
cfaverage	1.4	11/07/2002 10:10:32	◇
cfcnvt	1.45	11/07/2002 10:06:44	◇
cfview	1.5	11/07/2002 09:59:57	◇
cfbeta	1.4	11/07/2002 10:33:13	◇
cfsequence	1.4	11/07/2002 10:30:24	◇
Wind Chemistry Manager	1.8	11/07/2002 10:16:15	◇
cfcombine	1.8	11/07/2002 11:08:37	◇
gmanpre	6.153	11/07/2002 12:05:08	◇
gridvel	1.6	11/07/2002 12:40:00	◇
Madcap production	1.13	11/07/2002 12:31:21	◇
adfedit	1.5	11/07/2002 12:41:39	◇
Common File Class Library	1.5	11/07/2002 12:48:00	◇
cfpost_pre	3.163	11/05/2002 14:28:29	◇
cfsplit	1.7	2/04/2002 11:21:02	◇
Domain Decomposition	1.5	2/04/2002 13:11:37	◇
gpro	1.5	2/06/2002 08:02:35	◇
Tools Makefiles	1.18	11/05/2002 14:25:38	◇
Madcap production Library	1.10	11/05/2002 14:32:50	◇
jormak	1.4	11/07/2002 10:57:15	◇
tmptrn	1.7	11/07/2002 11:05:32	◇
etc.	...	...	◇

The Wind documentation can be found at http://www.lerc.nasa.gov/www/winddocs/

System Notes

Figure 4: Tools Projects Page.

2.3.1 Tools Application Distribution

To download the tools application distribution, on the Tools Projects Page (Figure 4) select "Tools Makefiles" by clicking on its 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 Tools Makefiles Download Page, the top part of which is shown in Figure 5. 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 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) and GMAN (gmanpre). 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. Another tool, MADCAP (Madcap production), does not appear on the list, but can also be downloaded separately.

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.

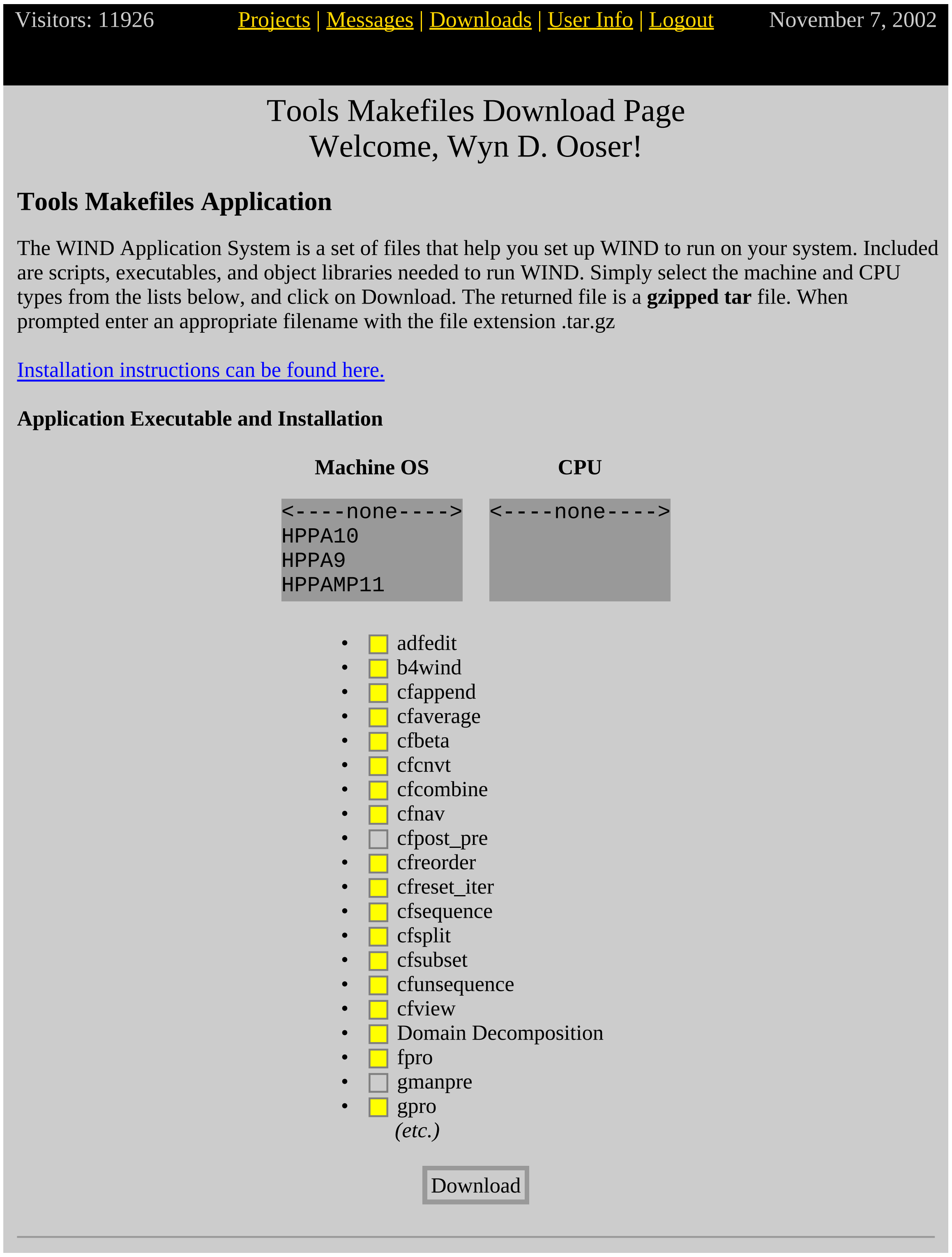
To download the tools build distribution, simply click on the "Here" button in the bottom part of the Tools Makefiles Download Page. IVMS will package the necessary files into a gzip'ed tar file, and, if the "Download Later" button is not selected, you'll be prompted for the file name, which should end 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 4) 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



Visitors: 11926 Projects | Messages | Downloads | User Info | Logout November 7, 2002

Tools Makefiles Download Page
Welcome, Wyn D. Ooser!

Tools Makefiles Application

The WIND Application System is a set of files that help you set up WIND to run on your system. Included are scripts, executables, and object libraries needed to run WIND. Simply select the machine and CPU types from the lists below, and click on Download. The returned file is a **gzipped tar** file. When prompted enter an appropriate filename with the file extension .tar.gz

Installation instructions can be found here.

Application Executable and Installation

Machine OS	CPU
<----none---->	<----none---->
HPPA10	
HPPA9	
HPPAMP11	

- ☐ adfedit
- ☐ b4wind
- ☐ cfappend
- ☐ cfaverage
- ☐ cfbeta
- ☐ cfcnvt
- ☐ cfcombine
- ☐ cfnav
- ☐ cfpost_pre
- ☐ cfreorder
- ☐ cfreset_iter
- ☐ cfsequence
- ☐ cfsplit
- ☐ cfsubset
- ☐ cfunsequence
- ☐ cfview
- ☐ Domain Decomposition
- ☐ fpro
- ☐ gmanpre
- ☐ gpro

(etc.)

Figure 5: Tools Makefiles Download Page.

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 "Here" button in the bottom part of the Download Page. IVMS will package the necessary files into a gzip'ed tar file, and, if the "Download Later" button is not selected, 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

It is suggested that one individual from an organization download the needed application distributions and be a single point of contact for the WIND software. If possible, the executables should be located in a central location that can be accessed via NFS in a consistent manner. This way, the location and method of accessing the code will be the same on all user machines, and the code can be updated without impacting the user significantly.

After downloading the gzip'ed tar files containing the application and tools distributions for the platforms to be used, the recommended installation procedure is as follows:

1. If previous versions of WIND and/or the WIND tools have been installed on the system, rename the old CFDROOT and TOOLSROOT directories (the directories themselves, that is, not the CFDROOT and TOOLSROOT environment variables), and check to be sure that these directories are *not* in your search path.[2]

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

3. In that directory, unpack the files by doing, for both files:

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

   where *filename* is the file name, including the *.tar.gz* extension. If application distributions were downloaded for several different machines, all of the gzip'ed tar files should be unpacked before proceeding to the next step.

4. Run the *INSTALL.appl* script. You'll be prompted for the directory into which the WIND scripts and executables should be placed. As noted above, it's convenient to create and use a directory that multiple users can access via NFS. The files will be copied into a *wind* subdirectory directly below the directory you specify. For example, responding with */usr/local/wind* will result in all the scripts and executables being copied into */usr/local/wind/wind*.

   If the scripts and executables are to be kept under the original directory used for the gzip'ed tar files in Step 2, simply respond to the prompt with "." (without the quotes).

   The *INSTALL.appl* script will not modify any of your system or user configuration files. It simply copies the appropriate files to the directory you specify. It also modifies two files included with the WIND application distribution — *cfd.login* and *cfd.profile* — to define the environment variable CFDROOT as the full path name of the directory specified above.

5. Run the *Install.tools* script. This step is exactly analogous to the one above, and is used to install the scripts and executables for the various pre- and post-processing tools used with WIND. The installation directory should be the same as specified in the previous step. The files will be copied into a *tools* subdirectory directly below the specified directory.

   As before, to keep the tools scripts and executables under the original directory used for the gzip'ed tar files in Step 2, simply respond to the prompt with "." (without the quotes).

[2]For those of you asking "Why?" — this will ensure that the new versions of the scripts can be properly installed, and that the new version of *pvmgetarg* will be used to determine the SYSTEM and SYSTEM_CPU environment variables. After the installation of the new version has been completed, earlier version(s) of WIND may be removed, or copied to the newly-installed *bin* directory under a different name.

As in Step 4, the *Install.tools* script modifies two files included with the tools application distribution — *tools.login* and *tools.profile* — to define the environment variable TOOLSROOT.

While that completes the actual installation of the WIND and tools application distributions, users must perform one additional step. *csh* and *tcsh* users should add the following two lines to their *.login* file in their home directory:

```
source dir_name/wind/bin/cfd.login
source dir_name/tools/bin/tools.login
```

where ***dir_name*** is the directory specified in Step 4 during the installation. Similarly, *sh* and *ksh* users should add the following two lines to their *.profile* file in their home directory:

```
. dir_name/wind/bin/cfd.profile
. dir_name/tools/bin/tools.profile
```

This will automatically modify the user's search path to include the locations of the WIND and tools scripts and executables, and define some additional environment variables used by some of the scripts.

Note that, to run WIND or any of the tools immediately after completing the above process, you (and any other users currently logged on) should logout and log back in to execute the *.login* or *.profile* file, setting some environment variables, including CFDROOT, SYSTEM, and SYSTEM_CPU, and modifying the PATH environment variable to include the newly-created locations for the WIND and tools executables. Alternatively, *csh* and *tcsh* users could execute

```
source .login
```

and *sh* and *ksh* users could execute

```
. .profile
```

in their home directory.

3.1.1 Variation for NFS Users

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 "central" system, a slight variation of the above procedure is necessary. The definitions of the environment variables CFDROOT and TOOLSROOT in the *cfd.login*, etc., files on the central system should be modified to point to 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.

For example, at NASA Glenn *automount* is normally used to access NFS file systems. The WIND and tools application distributions are stored on a central workstation (named, say, *wind_machine*), in the directory *dir_name* as specified in Step 4 of the previous section. This directory is exported read-only to the "home" system, normally an individual's workstation, for each WIND user.

The *cfd.login*, etc., files on *wind_machine* were copied to new files called *cfd.nfs.login*, etc.[3] 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. E.g., if *dir_name* is */usr/local/wind*, then the user automounts the *wind* directory by typing

```
cd /net/wind_machine/usr/local/wind
```

and the definitions of CFDROOT and TOOLSROOT in *cfd.nfs.login*, etc., are `/net/wind_machine/usr/local/wind/wind` and `/net/wind_machine/usr/local/wind/tools`, respectively.

Finally, using the above example, *csh* and *tcsh* users modify their *.login* file to add the lines

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

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

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

As in the previous section, to run WIND or any of the tools immediately after adding these lines, users currently logged on should logout and log back in to execute the *.login* or *.profile* file, setting the CFDROOT and TOOLSROOT environment variables. Alternatively, *csh* and *tcsh* users could execute

```
source .login
```

and *sh* and *ksh* users could execute

```
. .profile
```

in their home directory.

3.2 Installing on a Windows System

Unfortunately, due to resource limitations, the NPARC Alliance cannot officially support a Windows version of WIND. Nevertheless, application distributions for Windows may be available for WIND 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 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.

[3]This was done so that users logging directly into *wind_machine* could still access WIND as described in the previous section.

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 the WIND code.

4 Installing the Build Distributions

If pre-compiled executables for the WIND code 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 the WIND Code

Makefiles are distributed with the WIND build distribution for the following platform, operating system, and CPU combinations.

- Compaq Tru64 Unix multi-processor Alpha processors
- Convex
- Cray
- Hewlett-Packard
  - Exemplar CSPP, PA-8000 processor
  - HP-UX 9.*x*, PA-7200 processor
  - HP-UX 10.*x*, PA-8000 processor
  - HP-UX 11.*x*, multi-processing, PA-8500 processor
- IBM RS6000, single- and multi-processing RS64 and PC600 processors
- Linux, single- and multi-processing *x*86 processors
- Silicon Graphics[4]
  - IRIX 5.*x*, R4400 processor
  - IRIX 6.*x*, single- and multi-processing, R4400, R5000, R8000, R10000, R12000, and R14000 processors
- Sun SunOS 5.*x* (Solaris 7+), single- and multi-processing SPARC4U processors

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 5 for further information. The NPARC Alliance User Support Team (`nparc-support@info.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 application distribution (even if it doesn't include the pre-compiled executable), as described in Section 3. The WIND build distribution is a complete package, however, and, using the following procedure, a working system can be constructed from it alone.

[4]For parallel operation on SGI systems running IRIX 6.2 through 6.5, the WIND code being run on the master processor must be built using the MIPSpro 7.2 (or later) compilers. Beta runs have determined that if WIND (and PVM) on the master is built using the MIPSpro 7.1 compilers, all zones will (improperly) be assigned to whatever host is listed first in the *.mpc* file. If WIND (and PVM) on the master is built using the MIPSpro 7.2 (or later) compilers, everything works as expected, even if the WIND (and PVM) executable being run on the workers is built with the 7.1 compilers. Earlier versions of IRIX and the MIPSpro compilers have not been tested.

1. Put the gzip'ed tar file containing the WIND 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 source code.
3. Check to see if the CFDROOT, SYSTEM, and SYSTEM_CPU 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 "dir_name/wind-dev"
   #=ENDROOT=
   endif
   ```

   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="dir_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 dir_name/wind-dev/bin/cfd.login
   ```

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

   ```
   . dir_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, and to modify your PATH to include the location of the WIND 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 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 type of system/OS and CPU being used.[5]

   All the options in these files should be reviewed, but pay 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 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`.
   - *source/Makefile.user*
   - *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 definition of the Fortran compiler used in the `FC`, `F90`, and `LD` variables. Note all three require a Fortran 90/95 compiler.

[5] Several systems use a generic makefile and use links to create the specific file for your system. For example, on an SGI R12000 multi-processing system running IRIX 6.5, *Makefile.include.SGIMP6.5.R12000.opt* will be a link to *Makefile.include.SGI6.R12000.opt*.

  - 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.
  - The makefiles for SGI R12000 and R14000 systems include a `-r12000` option in the definition of `ABI`, which requires the MIPSpro 7.3 (or later) compilers. If you're using the 7.2 compilers, either remove the `-r12000` option, or change it to `-r10000`.
- *makefiles/pvm_conf/SYSTEM.SYSTEM_CPU.def.xxx*, where *xxx* corresponds to *opt*, *dbx*, or *pure*.
  - In the definition of `ARCHCFLAGS`, the `RSHCOMMAND` option sets the full path name of the "remote shell" command (normally *rsh*, *remsh*, or *ssh*) to be used for parallel operation using PVM message passing. It must be set correctly if WIND is to be run in parallel mode using PVM.
  - The makefiles for SGI R12000 and R14000 systems include a `-r12000` option in the definitions of `ARCHCFLAGS` and `ARCHFFLAGS`, which requires the MIPSpro 7.3 (or later) compilers. If you're using the 7.2 compilers, either remove the `-r12000` option, or change it to `-r10000`.

10. From the *wind-dev* directory, run *make* to create all the required libraries and executables for running the WIND code. 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 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 amd PVM executables. To do that, issue the commands

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

    The first two commands copy the WIND executable to *$CFDROOT/SYSTEM/CPU/bin*, and the WIND run scripts to *$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 platform/operating system and the CPU type, respectively. For example, if the code is being built on an SGI workstation running IRIX 6.5, with an R10000 CPU, *SYSTEM* and *CPU* will be *SGI6.5* and *R10000*, respectively.

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

4.2 Building the Tools

Makefiles are distributed with the tools build distributions for the the same platforms supported by the WIND build distribution. Unlike the WIND 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.[6] Thus, one could have separate directory trees for the WIND 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 the WIND solver, 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:

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

   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="dir_name/wind-dev/tools-dev"
   #=ENDROOT=
     export TOOLSROOT
   fi
   ```

[6] There is one minor exception to this, for CFPOST, described below in step 10.

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

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

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

   ```
   . dir_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 compiler 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*
   - *makefiles/Makefile.include.SYSTEM.SYSTEM_CPU.opt*
   - *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 platform/operating system and the CPU type, 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 WIND 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 build process. Please report problems to the NPARC Alliance support team at *nparc-support@info.arnold.af.mil* or (931) 454-7885.

5 Porting WIND to a New UNIX Platform[7]

5.1 Porting Guidelines

The first step to porting WIND is to obtain and install the WIND application distribution, which contains the scripts used to run the code. (See Section 2.2.1 and Section 3.1.) Even though there is no executable available to run, you still need the scripts, because the "make" system relies on them to determine what type of machine and CPU it is working on. After you have gone through the process described there, check to make sure that the CFDROOT, SYSTEM, and SYSTEM_CPU environment variables are set to values that make sense for your particular system. Note that it is not strictly necessary that SYSTEM_CPU be set. (For instance, the CRAY and CONVEX systems do not have any particular CPU associated with them at this time.)

The source files for WIND (and the non-system libraries it depends on) are obtained by downloading the build distribution. (See Section 2.2.2.) 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. 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, and whether or not to build PVM. In addition, this file defines the machine and CPU type for which WIND 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.

- *$WIND_DEV/makefiles/Makefile.include.$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_wind/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

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

examples. Other possible extensions are *.pure*, which defines the compilation configuration for use with the *Purify* debugging software package. Files ending in *.check* are for use with the *ftnchek* static debugger.

- *$WIND_DEV/makefiles/pvm_conf/$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. 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 1: If you wish to run WIND in parallel, it is vital that `-DRSHCOMMAND` point to the correct remote shell command (including path) for your system.

  Note 2: As before, there are other extensions besides *.opt*. See the SGI files for examples.

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.

5.2 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 other, 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*.

   I know that these files should be links. I'll get there some day.

2. *Can I cross-compile WIND 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 R4400, R5000, R8000, and R10000 machines. (R12000 too, but we haven't got one to try it with — yet.)

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, 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 *$WIND_DEV/makefiles/Makefile.include.$SYSTEM.$SYSTEM_CPU.opt* and set the `AWK` variable to `nawk`.

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

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

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

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

```
cd $WIND_DEV/libcfd_wind
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 *$WIND_DEV/pvm* subdirectories in a similar fashion. Once you are satisfied that the symbol is not from anything in the WIND distribution, you will have to poke around the system libraries to identify which ones should be added to the `EXTRALIBS` line in *$WIND_DEV/makefiles/Makefile.include.$SYSTEM.$SYSTEM_CPU.opt* (or *.dbx* etc.).

If the symbols are from within WIND, 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_wind/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_wind to compile, but now WIND itself is complaining. Where should I look?*

    As with *libadf* (see #10) and *libcfd_wind* (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. 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. (The various scripts should set these variables as needed).

If you still can't compile 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 in single processor mode (see #3).

14. *I've gone through the whole process you describe above (in Section 5.1), 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), IVMS assigns only "read" permission to the files it sends you. 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 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 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 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 `http://cgns.sourceforge.net` 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))".[8]

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.

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