The Parallel Applications Development Environment (PADE) User's Manual

Judith E. Devaney
Robert Lipman
Minwen Lo
William F. Mitchell
Computing and Applied Mathematics Laboratory

Mark Edwards
Charles W. Clark
Physics Laboratory

National Institute of Standards and Technology

PADE – Major Release 1.4
November 21, 1995

Abstract

This manual describes the use of the Parallel Applications Development Environment (PADE) developed at the National Institute of Standards and Technology (NIST). PADE is a flexible, customizable environment for developing parallel applications that use the Parallel Virtual Machine (PVM) message-passing library. It provides an integrated framework for all phases of development of a message-passing parallel application: editing, compilation, execution, and performance monitoring and enhancement. The PADE package consists of an intuitive graphical user interface, a suite of PVM utilities, and extensive documentation in PostScript, ASCII, and HTML formats. PADE is most useful with a heterogeneous virtual machine consisting of multiple architectures and/or multiple file systems where it aids the parallel developer by helping to keep track of which source files must be sent and compiled on which hosts, sending the files, issuing the compilation commands, and displaying the results of the compilation on the developer's terminal.

Some eminent scientists sprinkle their speech with foreign or foreign-sounding phrases, and in emulation of this practice we have adopted the pronunciation of PADE as "pah-DAY", which incidentally honors the French mathematician Henri Padé (1863-1953). Those who telephone NIST to ask how they can *get paid* may engender a misunderstanding.

Contents

List of Figures

Chapter 1

Introduction to PADE

This chapter provides an overview of the Parallel Applications Development Environment (PADE). It describes the philosophy behind PADE, the basic aspects of its functionality, and how to access PADE on the Internet.

PADE is a flexible, customizable environment for developing parallel applications that use the Parallel Virtual Machine (PVM) message-passing library. It provides an integrated framework for all phases of development of a message-passing parallel application: editing, compilation, execution, and performance monitoring and enhancement. PADE is most useful with a heterogeneous virtual machine consisting of multiple architectures and/or multiple file systems. By automatically selecting only those files that have been modified since the last compilation, PADE can be used as a *parallel make* for a heterogeneous virtual machine. The PADE package consists of an intuitive graphical user interface, a suite of PVM utilities, and documentation in PostScript, ASCII, and HTML formats.

1.1 The PADE Philosophy

The goal of PADE is to facilitate the development of parallel applications that run on a heterogeneous *virtual machine*, that is, a machine composed of independent networked computers (its *nodes*), which may be of different types, and/or different file systems.

Such an application will have components that reside on the file systems of different computers with different operating systems. These components will have to be compiled on each node, usually with differing compiler directives and options. The compiled versions must be run on each node, and the input and output streams must be managed on each node.

Consider the basic operations that arise in the development cycle of such a program:

1. **Editing** source files (potentially a different set of files for each node on the virtual machine).

2. **Transferring** sets of source files between nodes, as appropriate.

3. **Compiling** each set of source files on each node (in a *heterogeneous* environment, the compiler for each architecture, or even each node, may be slightly different).

4. **Executing** the updated program.

Steps two and three can become quite time consuming as the number of nodes increases, even if the variation of operating systems between nodes is relatively minor (for example, if they are different variants of Unix). Also, keeping track of which source files require editing in step one can also become quite cumbersome.

The basic concept of PADE is to provide a development console for the virtual machine, so that all essential parallel development tasks can be carried out at a central location. This central location, the *development host*, is the computer where all source files for the parallel application are kept and is where PADE is executed. PADE provides a graphical interface to the operations of the virtual machine. PADE allows the following tasks to be performed on the development host, with transparent access to the other nodes of the virtual machine:

- Defining the virtual machine, that is, identifying the computers that will run the various components of the parallel application
- Organizing the application source files
- Editing source code appropriate to each host while retaining all source files in a common location on the development host
- Building the parallel application with a single command, operations that are launched by this command include selectively sending files to selected nodes of the virtual machine (optionally only files that have been recently modified) and issuing the appropriate compile command on each node
- Reviewing compilation and execution output
- Executing the application on the virtual machine, with or without the aid of parallel debuggers and performance monitors
- Accessing online help:
  - this manual
  - introductory PVM man pages
  - the PADE World-Wide-Web (WWW) home page

A graphical interface is a useful aid for keeping track of the application, as it enables one to visualize how the various parts of the application are distributed across the virtual machine. The current release of PADE offers a graphical representation of the virtual machine, which shows the nodes and their associated program files on a two-dimensional diagram. Many of the file and command management actions that are required during application development can be carried out by familiar "click and drag" operations on objects in this diagram, which simplifies much routine administrative work that would otherwise distract the programmer's attention.

1.2 How PADE Works

Central to the operation of a program like PADE is a vehicle for passing messages between nodes, that is, a *message-passing library.* The current version of PADE is based upon the widely used Parallel Virtual Machine (PVM) message-passing library developed at Oak Ridge National Laboratory. Indeed, it is a PVM *application* in that it utilizes PVM calls for all communication between processors of the virtual machine, although it should be straight forward to implement PADE for another message passing library (such as MPI).

The PVM software enables a user to write applications in which a collection of computers perform parts of the application in parallel. This group of computers can be considered to be a *virtual machine*

(VM). Each computer on the VM will sometimes be referred to as a *node* and sometimes as a *host*. These two terms will be used interchangeably in this manual.

A VM can be *homogeneous*, that is, consist of identical processors sharing a common file system, or *heterogeneous*, that is, consist of more than one architecture or more than one file system. PADE is most effective in aiding parallel development on the latter type of virtual machine where consistency must be maintained between files in different locations.

A working PVM application consists of a set of files containing C and/or FORTRAN source codes, makefiles, data, etc. Some subset of these files will be compiled on a given host of the VM to create a binary that is the component of the entire PVM program for that host. The subset of files needed to create this executable may vary from host to host. This may be the case for several reasons some of which include:

- the portion of the task to be performed by the application allotted to a particular host is different from another for load-balancing reasons (some hosts may be faster than others)
- non-portability of the code
- host-specific code optimizations
- a different makefile is required for compilation on some hosts since there may be different compiler names, library paths, and include paths

The creation of the binary on a given host will require the transfer of files to that host and the launching of the compile command.

The two ways in which PADE aids the parallel developer are by helping to keep track of which source files must be sent and compiled on which hosts, and by actually sending the files and issuing the compile commands. In the PADE development model the developer keeps all source files on a single host (the *development host*) and provides the names of the hosts, the files needed to build the binary on each host, and the commands required to build the binary on each host. In the rest of this manual, this information is referred to as the *PADE data*.

Once the PADE data have been entered, the developer begins the edit/transfer/compile/execute cycle. In the edit phase, PADE enables the developer to modify any file described in the PADE data information with a user-specified editor. PADE enables the developer to perform the transfer and compile phases in one step using *PVMmake* (provided in the PADE distribution). PADE also displays the results of the compilations on the various hosts in a display window that can be scrolled through at leisure and saved if desired. In the execution phase, PADE enables the developer to launch the application being developed and to view the output in a display window.

1.3 How to Get PADE

The PADE package can be obtained by anonymous ftp at
ftp://math.nist.gov/pub/pade
in the file **nist_pade.1.4.x.tar.gz** where the integer x denotes the number of updates to release 1.4.0.
Current information on PADE is maintained at the PADE World Wide Web home page at
http://www.itl.nist.gov/div895/pade/pade.html
which also contains links to all required software packages and an HTML version of the PADE User's Guide. Questions, comments and bug reports can be sent by electronic mail to `robert.lipman@nist.gov`.

1.4 Changes Since Version 1.0

The major changes to PADE in Version 1.4 are:

- Support for Tcl/Tk 8.0 (older versions of Tcl/Tk no longer supported)
- Enhanced capability to selectively send files to remote hosts
- New file browser
- Modification of File Browser Default Path preference

The major changes to PADE in Version 1.2 are:

- Added capability to selectively send files to remote hosts
- Moved functions of the `Setup` window (adding hosts and files, opening and saving PADE files) to the Main window under the `File` and `Setup` menus and `Graph` button
- Renamed the Tree View to Graph
- Moved functions of the `Run` menu to the `PVMmake` menu and `Apps` menu
- Moved functions of the `Monitor` menu to the `Apps` menu
- Moved function of setting preferences under the `Help` menu to the `Preferences` menu under the `File` menu
- Moved function of the `Exit` button to `File` menu
- Added `Undo` and `Sort` buttons
- Modified hosts and files manipulation capabilities in the Graph to have the same capabilities as those under the `Setup` menu
- Added a .paderc file to save user preferences and other information
- PADE can now be invoked with the name of a PADE data file on the command line
- Made the "look and feel" of the graphical user interface more "standard", for example: `File` menu, accelerator keys, pasting selections into entry windows, double-clicking to pick items in a listbox
- The *PVMmake* configuration file is no longer automatically written. It must be saved under the `PVMmake` menu
- Removed NIST-specific directories in *pvm_tidy*
- Added -ep option to padeinituser, to specify what directory to initialize
- Major revision of the Installation and User's Manuals

1.5 Known Bugs

PVMmake is not currently capable of transferring binary files. In order to transfer a binary file, it must first be converted to ASCII by *uuencode* or a similar program.

Chapter 2

Preparing to Use PADE

This chapter provides instructions for preparing your environment for using PADE. This consists of setting several environment variables, adding some directories to the executables path, preparing a PVM subdirectory, and creating a *.rhosts* file.

In this chapter it is assumed that all required software has been installed, including PVM and PADE. See the *PADE Installation Guide* for instructions to install PADE, or consult your system administrator to find the directories where PVM, PADE, and other software packages are installed.

2.1 Environment Variables and Path

There are several environment variables that must be set before PADE will run. Also, there are some directories to be added to the executables path. This is most easily accomplished by adding a few lines to the non-login shell startup file (*.cshrc, .profile, .bashrc*, etc.). The environment variables on the development host are:

- PADE_ROOT – the directory in which PADE is installed
- PVM_ROOT – the directory in which PVM is installed
- PVM_ARCH – the PVM name of your architecture
- XPVM_ROOT – the directory in which XPVM is installed (optional)
- TCL_LIBRARY – the directory in which the TCL library resides
- TK_LIBRARY – the directory in which the TK library resides
- EDITOR – your favorite editor (optional; PADE will prompt if not set)

The additional directories to be added to the executables path on the development host are:

`$HOME/pvm3/bin/$PVM_ARCH`
`$XPVM_ROOT/src/$PVM_ARCH` (optional)
`$PADE_ROOT/caml_utils/$PVM_ARCH`
`$PADE_ROOT/pade`
`$PVM_ROOT/lib/$PVM_ARCH`
`$PVM_ROOT/lib`

The `caml_utils` directory is not necessarily installed under `$PADE_ROOT`. Consult your system administrator for the actual location.

On the remote hosts, the user is only required to have the environment variables PVM_ROOT and PVM_ARCH. None of the path additions are required on the remote hosts.

For further information on these environment variables and directories, see the script *pade_setenv* (if your shell is csh or tcsh), or *pade_export* (if your shell is sh, ksh or bash) in the directory `$PADE_ROOT/examples`. An alternative to entering this information in the shell startup file is to execute one of these scripts before starting PADE.

2.2 PVM Directory

The directory `$HOME/pvm3/bin/$PVM_ARCH` is where PVM looks for executable files for spawned processes by default. If you do not use the "ep=" option in the PVM hostfile, this directory must exist, and must contain symbolic links to some of the programs in the `caml_utils/$PVM_ARCH` directory. This directory structure must be present on all hosts, development and remote. A utility is provided to create the required directory structure if it does not exist. It must be run on each architecture/file-system combination. The executable for this utility is *padeinituser*, and it is located in the `caml_utils/$PVM_ARCH` directory. If you do not use the "ep=" option, then run

```
padeinituser
```

to initialize the default directory `$HOME/pvm3/bin/$PVM_ARCH`. If you use the "ep=" option in the PVM hostfile, then one of the directories specified by ep must contain symbolic links to some of the CAML Utilities. This can be accomplished by

```
padeinituser -ep dir
```

where `dir` is a directory specified by "ep=" in the PVM hostfile.

2.3 The *.rhosts* File

For PADE to function properly, a program on one machine must have permission to write into the file system of another machine. This is accomplished through the file *.rhosts* in the user's home directory. This file must reside in every file system of the virtual machine, and must list every host name in the virtual machine.

Chapter 3

Running PADE: A Tutorial

PADE is intended to be an intuitive graphical user interface for developing parallel applications on a virtual machine. The typical use of PADE consists of the following steps:

1. Initiate the virtual machine using the `PVM` button.
2. Construct the PADE data using the `Setup` button or `Graph` button.
3. Distribute and compile the application using the `PVMmake` button.
4. Run the application using the `Apps` button.
5. Edit files, redistribute modified files, and repeat the cycle as necessary.

In this chapter we go through an example of this process using the *manager/worker* program in the `examples` subdirectory of the PADE distribution. This program is a variation of one of the examples contained in the PVM distribution. The files located in the `examples` subdirectory are:

- *manager.c* – A slightly modified version of the *master.c* program provided with the PVM software.
- *worker.c* – A slightly modified version of the *slave.c* program provided with the PVM software.
- *makefile.mw* – A generic makefile that can be used to compile the *manager.c* and *worker.c* source files. A modified copy of this makefile will be needed for each architecture/file-system combination of the user's virtual machine.
- *pade_setenv* – A script that can be executed to set the environment variables and directory paths that are needed in order for PADE to run. This file assumes the user is running either the csh or tcsh shell.
- *pade_export* – A script similar to *pade_setenv* except that it assumes the user is running either the sh, ksh, or bash shell.
- *mw* – An example PADE data file used to compile the source files.
- *mw.pvmmake* – The *PVMmake* config file corresponding to the *mw* PADE data file.
- *pvmmake.output* – An example of the output from running *PVMmake* to compile the *manager/worker* example.

- *mw_make_clean* – An example PADE data file used to remove the executable and object files created on all virtual machine nodes during compilation of the *manager/worker* example program.
- *mw_make_clean.pvmmake* – The *PVMmake* config file corresponding to the *mw_make_clean* PADE data file.
- *mw_run* – An example PADE data file used to execute the *manager/worker* program using *PVMmake*.
- *mw_run.pvmmake* – The *PVMmake* command file corresponding to the *mw_run* PADE data file.

The *manager.c* and *worker.c* source files need not be modified by the user. The user will need to make several copies of *makefile.mw*, possibly one for each node on the virtual machine, and modify each appropriately. The PADE data files and corresponding *PVMmake* files are provided only as examples and will not be needed by the user.

3.1 Example PADE Session

In this section a PADE session is followed step by step. It is assumed that the PADE software is installed as described in the *PADE Installation Guide*, and that the user has configured his or her environment as described in Chapter 2 of this manual. Each of the following subsections contains a brief explanation of a step of the PADE session and the specific actions you should take.

Before beginning the tutorial, you should copy the `examples` subdirectory to another directory and create several copies of the makefile. You will need one makefile for each architecture/file-system pair in your virtual machine. Put each makefile in *makefile.hostname* where hostname is the name of one of the machines with the given architecture/file-system, and modify the makefile as needed.

3.1.1 Starting PADE

PADE is invoked by typing *pade* at the command prompt. PADE will first read environment variables and then search the path for the stand-alone programs that it can potentially execute. The user should see the following message in the window in which PADE was invoked:

```
Welcome to PADE (1.4)
Checking environment variables
Checking commands
```

PADE will then proceed with a list of the stand-alone programs that it was unable to find by searching the directories in the user's PATH, and environment variables that it was unable to find. After this, the PADE main window (Figure 5.1) should appear.

1. **Type *pade* at the command prompt.**

3.1.2 Starting the Virtual Machine

The first action the user should take after invoking PADE is to create a PVM hostfile and start the PVM console. The user is strongly encouraged to keep the pop-up window containing the PVM console open as long as PADE is running. One reason for this is that PADE will hang if the user executes a parallel application that hangs. This can be remedied by halting the VM from the PVM console.

1. **Select `Edit Hostfile...` under the `PVM` button.**

2. **Enter a file name (for example, *pvmhostfile*) in the pop-up `PADE Edit File` window, and click on the editor button** (the editor button will be labeled with the editor specified by the user for example, *vi*).

3. **In the pop-up editor window, create a file containing the host names in the virtual machine, one name per line.**

4. **Select `PVM console` under the `PVM` button.**

5. **Type *conf* at the PVM prompt in the pop-up PVM console window to view the configuration of your virtual machine.**

3.1.3 Entering Host Names

The first part of the set-up process is to construct the PADE host list. Hosts may be added to the list one at a time using `Add Host...` under the `Setup` button, however it is quicker to obtain the host list directly from the PVM hostfile as described above. It is crucial that the host names be *identical* to those known to PVM, as indicated by the PVM console command *conf.* A common example of when these names may differ is that PVM may drop the domain from the name of the development host.

1. **Select `Add from Hostfile ...` under the `Setup` button.**

2. **Double-click on the name of the PVM hostfile you previously created.**

3. **If any of the host names in the PADE main window are not *identical* to those in the PVM console window, then**

   (a) **Click on the host name in the PADE main window.**
   (b) **Select `Modify Host` under the `Setup` button.**
   (c) **Edit the host name in the pop-up `PADE Modify Host` window.**
   (d) **Click on `Apply`**

3.1.4 Entering Host Commands via Setup

Now that all the hosts have been entered, the user can proceed to enter the host commands. The commands to be entered for a particular host will be those necessary to compile the *manager.c* and *worker.c* programs. In this example, the general procedure for compiling these programs will be to change to the remote-host directory in which the source files are located and to issue the "make" command. The makefile appropriate to the particular host will have already been sent as part of the file-transfer phase of the process of building the parallel application.

1. **Click on the first host name in the PADE main window.**

2. **Select `Modify Host` under the `Setup` button.**

3. **Enter the command**
   `cd <path to your home directory>/pvm3/bin/ARCH; make`
   in the `Host commands:` part of the pop-up window, where `ARCH` is the PVM architecture name for that host.

4. **Click on `Apply`.**

3.1.5 Entering Host Commands via Graph

An alternative approach to entering host commands is to use the PADE Graph.

1. **Click on the Graph button.**
2. **Select Modify under the Hosts button in the PADE Graph window.**
3. **Click on the circle representing the second host.**
4. **Enter the command**
   cd <path to your home directory>/pvm3/bin/ARCH; make
   in the Host commands: part of the pop-up window, where ARCH is the PVM architecture name for that host.
5. **Click on Apply.**
6. **Repeat for remaining hosts.**
7. **Click on Close in the PADE Graph window.**

At this point, you should enter the host commands for all hosts using either the Setup or Graph approach. Note that if two hosts have the same architecture and share a common file system, only one of them needs the compilation commands.

3.1.6 Entering Files via Setup

The names of the files that must be transferred from development host to remote host must be entered for each remote host. In this example, three files will be transferred to each host. They are *manager.c*, *worker.c*, and *makefile.**, where the makefile name extension indicates for which host the makefile is appropriate.

The following process shows three ways to enter file names: 1) use the file browser, 2) copy and modify an existing file name, and 3) type in the file name using path names from prior entries. When typing in file names, it is possible to use the "*" symbol as a wild card. For example, *.c will create an entry for each file with the suffix "c".

1. **Click on the first host name in the PADE main window.**
2. **Select Setup Files... under the Setup button.**
3. **Click on the Add button in the PADE Files window.**
4. **Click on the File Browser button in the pop-up window.**
5. **Double-click on *manager.c* in the file browser.**
6. **Type <full path of your root directory>/pvm3/bin/ARCH/manager.c in the Remote file: part of the PADE Add File window.**
7. **Click on Apply and Close.**
8. **Click on the *manager.c* line in the PADE Files window.**
9. **Click on Copy in the PADE Files window.**

10. **Click on `Modify` in the `PADE Files` window.**
11. **Change *manager.c* to *worker.c* on the `Local file:` and `Remote file:` lines of the `PADE Modify File` window.**
12. **Click on `Apply`.**
13. **Click on `Add` in the `PADE Files` window.**
14. **Click on `Fill with last path` in the `Local file:` part of the `PADE Add File` window.**
15. **Type the name of the makefile to send to this host.**
16. **Click on `Fill with last path` in the `Remote file:` part of the `PADE Add File` window.**
17. **Type `makefile`**
18. **Click on `Apply` and `Close`.**
19. **Click on `Close` in the `PADE Files` window.**

3.1.7 Entering Files using Graph

An alternate means of creating the file list is to use the PADE Graph. The following process shows two ways to copy the file list from one host to another. Steps 1 through 9 illustrate copying files one at a time from the first host to the second host. Steps 10 through 13 illustrate copying files all at once from the first host to the third host.

1. **Click on `Graph` in the PADE main window.**
2. **Select `Copy` under the `Files` button of the `PADE Graph` window.**
3. **Click on the *manager.c* rectangle of the first host.**
4. **Repeat for *worker.c* and *makefile*.**
5. **Drag the newly created *manager.c*, *worker.c*, and *makefile* rectangles from the first host to the second host, and drop them on the circle.**
6. **Select `Modify` under the `Files` button.**
7. **Click on the *makefile* rectangle of the second host.**
8. **Change the name of the makefile under `Local file:` to the makefile for the second host, and change the path to have the correct ARCH under `Remote file:`, and click on `Apply`.**
9. **In the same way, modify the path for the `Remote file:` for *manager.c* and *worker.c***
10. **Select `Copy From Host` under the `Files` button.**
11. **Click on the circle of the first host, and then on the circle of the third host.**
12. **Change the name of the makefile and paths for the third host as described above.**
13. **Repeat for remaining hosts and click on `Close`.**

At this point, you should enter the file list for all hosts using either the `Setup` or `Graph` approach. Note that if two hosts have the same architecture and share a common file system, only one of them needs to receive the files.

3.1.8 Saving the Setup Information

Once all of the hosts, host commands, and files have been entered to your satisfaction, the information can be saved. The first action saves the information in a file for PADE, and can be loaded into another PADE session. The second action creates the *PVMmake* config file (see Chapter 4) which contains the commands for *PVMmake*.

1. **Select `Save` under the `File` button in the PADE main window.**
2. **Type a file name, for example *my_mw*, in the `File name:` line of the file browser, and click on `OK`.**
3. **Select `Save Config File as...` under the `PVMmake` button.**
4. **Type a file name, for example *my_mw.pvmmake*, in the file browser, and click on `OK`.**

3.1.9 Running *PVMmake*

With the PADE data entered, you are now ready to distribute the files and compile the *manager/worker* example program. It is always a good idea to list the *PVMmake* config file to verify that your PADE data is correct. The format of the *PVMmake* config file is given in Chapter 4. The output of *PVMmake* can be displayed in either a terminal window (*xterm*, or a type specified by user preference) or a scrollable/savable window, depending on which form of the `Run` button you choose. For comparison, an example of this output can be found in the file *pvmmake.output* in the `examples` subdirectory.

1. **Select `List Config File` under the `PVMmake` button.**
2. **After reviewing the file, click on `Close`.**
3. **Select `Run (in PADE window)` under the `PVMmake` button.**
4. **After viewing the output in the pop-up window, click on `Close`.**

3.1.10 Running the Application in a Terminal Window

Once the *manager/worker* program has compiled successfully, the next order of business is to run it. This can be done within PADE in two different methods: in a terminal window, or by *PVMmake*. For the remainder of this section, we will assume that the terminal is of type *xterm*. The *xterm* method is useful when the application requires interactive input at the terminal and when the user does not wish to save anything the application writes to standard output. The *PVMmake* method is useful if the output of the application that is written to the screen is important to the user and if the input to the program can be redirected.

1. **Select `Add...` under the `Apps` button.**
2. **Type *manager* in the pop-up window, and click on `Done`.**
3. **Select `manager` under `Run (in xterm)` under the `Apps` button.**

4. **When you are prompted in the *xterm* window for the number of slave programs, enter a reasonable integer, for example 10.**

5. **Type *exit* at the shell prompt in the *xterm* window.**

3.1.11 Running the Application Using *PVMmake*

To run the parallel application using *PVMmake*, you create a new PADE data configuration with the development host as the only host, no files to transfer, and the command being what is required to run the application. Interactive input must be redirected from a file. The output from the pop-up window can be saved to a file with the `Save` button.

1. **Select `New` under the `File` button to delete the current PADE data.**

2. **Select `Add Host...` under the `Setup` button, and enter the development host name and the command**
   `cd $HOME/pvm3/bin/$PVM_ARCH; manager < manager.input`

3. **Save the new PADE data file and *PVMmake* config file with a new name, for example, *my_mw_run*.**

4. **Create a file called *manager.input* in `$HOME/pvm3/bin/$PVM_ARCH` containing just an integer, for example, 10.**

5. **Select `Run (in PADE window)` under the `PVMmake` button.**

3.1.12 Exiting PADE

1. **Select `Exit` under the `File` button in the PADE main window.**

3.2 Other *PVMmake* config files

It is useful in the development process to create and use *PVMmake* config files for other purposes. An example of this is a "make clean" config file. A sample of such a config file is provided in the `examples` subdirectory under the name *mw_make_clean* . This config file contains commands instructing *PVMmake* to issue "make clean" commands on each host. Users who inspect the sample makefile provided with the PADE distribution will notice a "clean" directive at the bottom.

Many other routine functions can be automated in a way similar to the "make clean" example. In this sense, PADE can be used creatively for more tasks than just parallel program development.

Chapter 4

PVMmake

This chapter discusses the function and operation of *PVMmake*, the utility that mediates all communications made by PADE between the development host and other nodes of the virtual machine. The actions of *PVMmake* usually take place without the need for attention by the user. However, if there is a problem during the file-transfer or command-execution operations performed by *PVMmake*, an understanding of the (very simple) *PVMmake* command syntax is essential for determining the problem.

4.1 Capabilities and Software Components of *PVMmake*

PVMmake is a PVM application, which uses the PVM message-passing library to perform two basic operations:

- **File transfer** – *PVMmake* can send an arbitrary text (ASCII) file from one arbitrary host to another arbitrary host.
- **Command execution** – *PVMmake* can issue an arbitrary shell command on an arbitrary host. Thus *PVMmake* can execute Unix commands and shell scripts as well as compiled programs. Furthermore, the output of these commands, scripts, or programs can be piped back to be viewed on the node where *PVMmake* was launched.

The fact that *PVMmake* performs just these two basic operations, makes the program very flexible, powerful, and very easy to use. PADE can be viewed as being essentially a shell for running *PVMmake*.

4.2 Running *PVMmake*

When the *pvmmake* binary is launched from the command line, it launches a copy of the *targets* binary on each node of the virtual machine and then reads and performs all of the commands listed in the *PVMmake config file*. Note that if *PVMmake* cannot start the *targets* program on *all* nodes of the virtual machine (which must already be running), it prints an error message and exits.

The syntax for starting *PVMmake* is as follows

```
pvmmake [-f pvmmake_config_file]
```

If *PVMmake* is launched without the optional file name parameter, then the program looks for a default config file named *pvmmake.config*. If it does not find this file, *PVMmake* prints an error message and exits. PADE essentially works by processing the information entered in the `Setup` window and writing a *PVMmake* config file. The user can then launch *PVMmake* to execute the commands contained in this file.

4.3 The *PVMmake* config file

The *PVMmake* config file can be created with an ordinary text editor. Thus *PVMmake* can be used in parallel applications development by those users who do not have access to the X Window System. The syntax of commands in the *PVMmake* config file is quite simple and is best illustrated by an example. An example config file, *mw.pvmmake* , is contained in the `nist_pade/examples` subdirectory. This file was written by PADE and comes from the *manager/worker* example.

The config file contains three types of items: 1) comments, 2) file transfer command, and 3) commands to be issued to remote hosts.

Lines beginning with the # character are comments.

The syntax for the *PVMmake* file-transfer command consists of two elements. These elements are

- One line containing the full path name of the local-host file to be transferred.
- One or more lines each containing two items: (1) the name of a host to which the file must be sent and (2) the file's location on that host. These two items must be separated by whitespace. This list of remote hosts/locations must be terminated by a **blank line**. A comment *cannot* take the place of the blank line.

Formally, the syntax is defined by:

```
file_name_on_local_host
remote_host_1      filename_on_remote_host_1
remote_host_2      filename_on_remote_host_2
...                ...
remote_host_N      filename_on_remote_host_N
<BLANK LINE>
```

Note that *local-host* refers to the computer on which the *pvmmake* binary was launched; *remote-host* refers to any node of the virtual machine.

The syntax for telling *PVMmake* to issue a command on a remote host has the following form:

```
@ remote_host command_1; command_2;...command_N [+errors]
```

The first character on the line must be the @ symbol. This is followed by the name of the host on which the command(s) are to be issued which is, in turn, followed by one or more commands separated by semicolons. An optional argument `+errors` is used to tell *PVMmake* to send any output generated by the commands issued back to the host on which the *pvmmake* binary was launched.

Chapter 5

The PADE User's Reference

This chapter contains a description of the PADE graphical user interface. Users who prefer to jump in and try a program will probably want to start with Chapter 3 in which can be found a step-by-step tutorial using the *manager/worker* example program that comes bundled with the PADE source code. Each time a button or menu is encountered which is unfamiliar to the user, it can be looked up in this chapter.

Before running PADE, all required software must be installed, several environment variables must be set, and a few directories must be added to the user's executables path. See the *PADE Installation Guide* for instructions for software installation, and Chapter 2 in this manual for setting the user's environment.

5.1 Starting PADE

PADE is invoked with the command

`pade` [*filename*]

The *filename* is optional; if present it must be the name of a PADE data file saved during a previous session.

If the file *.paderc* is present in the user's home directory, PADE will read this file to set some internal variables. The *.paderc* file is written at various times during the execution of PADE. The user is **not** required to create or edit this file outside PADE. The *.paderc* variables are:

- **CONFIRM** – either On or Off for confirmation dialogs; set in the `Set Preferences` menu (see Section 5.4).
- **FILE_PATH** – the directory where the file browser comes up; set in the `Set Preferences` menu.
- **LAST_PVMMAKE** – time when *PVMmake* was last run; set whenever *PVMmake* is run from PADE.
- **TERM** – user's favorite term command, such as *xterm*; set in the `Set Preferences` menu.
- **WWW** – user's favorite WWW browser command; set in the `Set Preferences` menu.
- **APP***n* where *n* is an integer – applications that can be launched from PADE; set in the `Apps` menu (see Section 5.11).

Menus use the left mouse button. Menus can be torn off by selecting the dashed line so that they do not disappear after making a selection. This opens a new window containing the menu, which can be closed or terminated in the usual way for your window manager.

There are accelerator keys for some of the menus. For example, Cntl-o opens a new file (see Figure 5.2). All menu items that have an accelerator key have the control sequence next to the menu item.

Buttons are disabled for any actions that cannot be performed, either because of missing optional software (for example, *XPVM*) or because of the current configuration state (for example, *PVMmake* cannot be run until the *PVMmake* config file has been selected).

5.2 Auxiliary Windows

This section describes several auxiliary windows that appear from multiple places within PADE.

- `File Browser` – used to select a file. A list of files in your home directory or the directory specified by `Set Preferences` is presented. Double-clicking on a file name selects that file. You can change to another directory by typing in a new directory name and clicking on `Rescan`, and create a new file by typing in a new file name.
- `PADE Edit File window` – accesses a file for editing. The file name can be entered in the text part of the window; usually a default file is provided by the parent window. There is a button to bring up the file browser to select a different file. There is a button to launch the user specified editor command.
- `PADE List window` – a scrollable/savable window used for listing files or output. The contents can be saved to a file by clicking on the `Save` button, or discarded by clicking on `Close`.
- Text entry boxes – many windows have a section for entering text. In most cases text can be pasted in from other non-PADE windows. Mark text in the other window by dragging the left mouse button over it. Paste it in the text entry box by clicking on the middle mouse button.
- Data entry windows – the pop-up windows for entering the PADE data under the `Setup` and `Graph` buttons contain an `Apply` button at the bottom. This *must* be pressed to apply the changes to the PADE data. When entering file names, the "*" character can be used as a wildcard.

5.3 The PADE Main Window

The PADE main window is shown in Figure 5.1. The nine buttons will be described in subsequent sections of this chapter. The two scrollable windows show the current PADE data. The left window lists the host names in the PADE data, and the right window shows the commands that *PVMmake* will send to those hosts.

5.4 The `File` Button

The `File` button, illustrated in Figure 5.2, is used to read and write PADE data files, set user preferences, and exit PADE. In all cases, if the current PADE data have been modified, the user will be prompted to save it before loading new data into pade. The menu entries are:

- `New` – clears the current PADE data. This is used to remove the PADE data before creating new data.

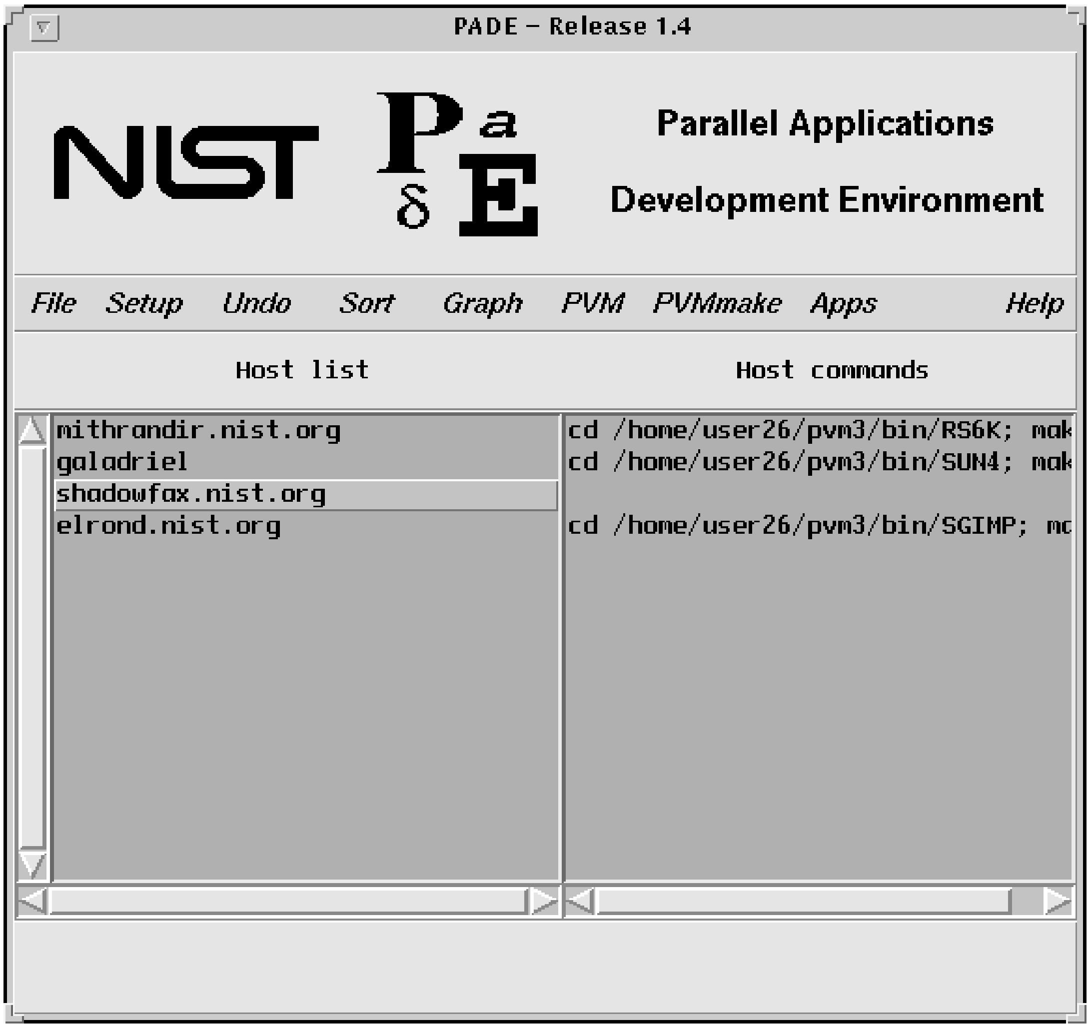

Figure 5.1: The PADE main window

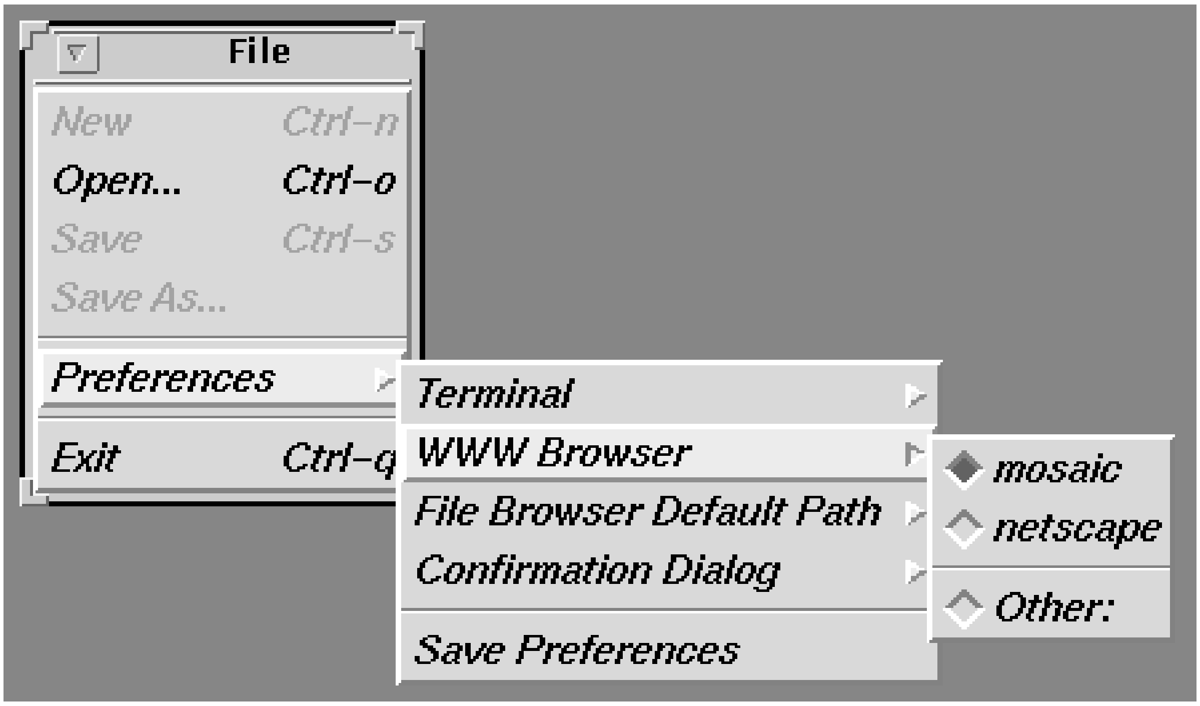

Figure 5.2: The File menu

- `Open...` – load a PADE data file into PADE. This is used to retrieve saved data. It opens a `File Browser` window (see Section 5.2).
- `Save` – write the current PADE data to the current PADE data file. This is used to update a file with the current PADE data. If a PADE data file is not open, a `File Browser` window pops up.
- `Save As...` – write the current PADE data to any file name. This is used to save the current PADE data in a new file. It opens a `File Browser` window.
- `Preferences` – sets user preferences. This has a submenu with the following entries:
  - `Terminal` – the kind of term window that is used for running *PVMmake*, applications, and some CAML Utilities, for example, *xterm*.
  - `WWW Browser` – the user's favorite World Wide Web browser command.
  - `File Browser Default Path` – pathname of the directory where the file browser will look for files.
  - `Confirmation Dialog` – toggles on or off the appearance of confirmation dialog windows whenever an action is taken.
  - `Save Preferences` – writes current values of the above preferences to the *.paderc* file in the user's home directory.
- `Exit` – end the PADE session.

Figure 5.3: The Setup menu

5.5 The `Setup` Button

The `Setup` button (Figure 5.3) is used to enter the PADE data in a textual manner. Alternatively, see Section 5.8 for how to enter these data graphically. The PADE data consist a list of hosts, a set of files to be sent to those hosts, and a set of commands to be executed on those hosts.

Most of the entries in this menu are activated by first clicking on a host name and then selecting the menu entry. The menu entries are:

- `Add Host...` – adds a new host to the list of hosts. It pops up a text entry box in which the user can enter a host name and a semicolon separated list of host commands. The `Last Command` button duplicates the command entered for the previous host. When *PVMmake* is run within PADE, the host will execute the commands.

- `Add from Hostfile...` – copies the complete list of hosts from the PVM hostfile, rather than entering them one at a time with the `Add Host...` entry. Note, however, that the host names must match those shown by the *conf* command in the PVM console window (Section 5.9) *exactly*, including the presence or absence of a domain name. You may need to modify some of the host names (see `Modify Host` below). This entry will pop up a file browser window from which you select a PVM hostfile.

- `Copy Host` – duplicates a host entry, including the list of commands and files. Since the list of commands and files is often nearly identical for different hosts, a "copy and modify" approach is often the quickest way to create new hosts in the PADE data.

- `Modify Host` – modifies a host name or commands. A text entry box will pop up with the current host name and commands, which can be modified.

- `Delete Host` – removes a host, its commands, and its files from the PADE data.

- `Setup Files...` – creates the list of files to transfer to the host. The `PADE Files` window will appear. This window is described in the next section.

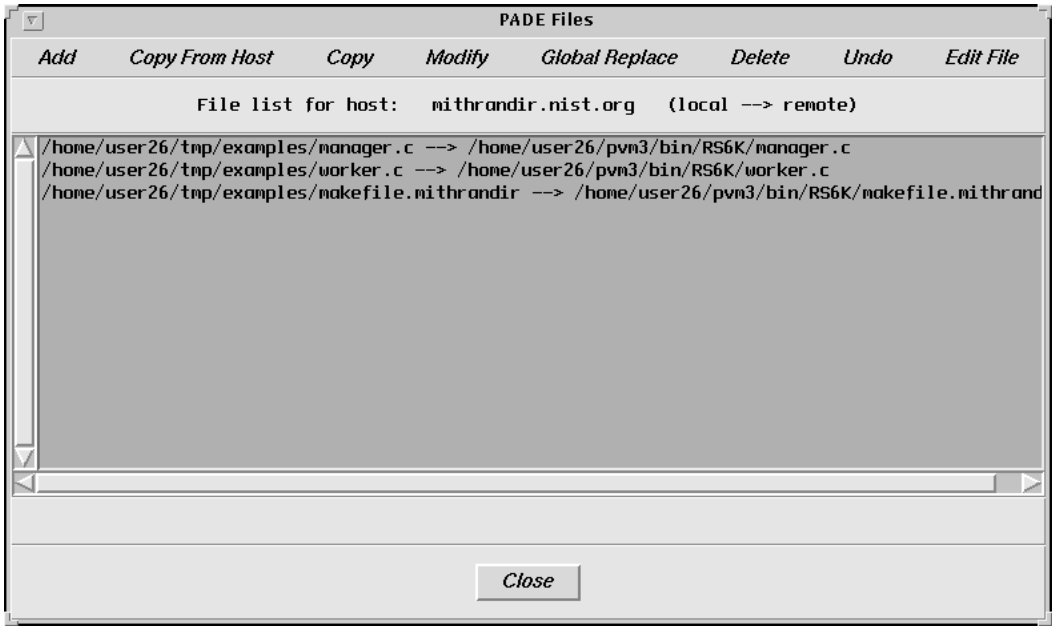

Figure 5.4: The PADE Files window

5.6 The PADE Files Window

The PADE Files window (Figure 5.4) is brought up by selecting Setup Files... under the Setup button, or by double-clicking on a host name in the PADE main window. This window is used to set the list of files to copy from the development host to another host. The name of the other host is shown near the top of the window.

The scrollable window shows the current list of files to be transfered to this host, including the full pathname on the development host and the full pathname on the destination host. The buttons in this window are used to create and modify these data.

- Add – adds a new file to the list. This button pops up a text entry window in which you type the full pathname of the file on the development host, and the full pathname of where to place the file on the destination host. On the development host, the file can be selected with the file browser. For either host, the Fill with last path button can reduce typing by providing the pathname of the last file added.

- Copy From Host – copies the file list from another host. Often a "copy and modify" approach is the fastest way to create a file list. Once the list is copied, the file designations can be changed using the Modify or Global Replace buttons (see below). This button pops up a window in which you select the host from which to copy the file list. Either single-click on the host name and on the Apply button, or double-click on the host name. Selecting Local only rather than the default Local and remote will create a list in which only the development host part of the entries is copied.

- Copy – copies an entry in the file list, which can subsequently be modified. First click on an entry in the list, and then this button.

- Modify – modifies an entry in the file list. First click on an entry in the list, and then this button. This will pop up a text entry window containing the selected entry, which can then be modified.
- Global Replace – replaces a string of characters by another string in all pathnames and file names in this file list. This is particularly useful when the Copy From Host button was used to create the file list, and the pathname is different on the other host. Clicking on this button pops up a text entry window in which you enter the two character strings. Clicking on the Last Replace button will initialize the strings with those used in the last global replace.
- Delete – removes an entry from the file list. First click on an entry in the list, and then this button.
- Undo – identical to the Undo button in the PADE main window (Section 5.7).
- Edit File – edits files on the development host. First click on an entry in the file list, and then this button. This will pop up a PADE Edit File window (Section 5.2).

5.7 The Undo and Sort Buttons

The Undo and Sort buttons are the only buttons in the main window that do not pop up an associated menu or window. The Undo button is used to cancel the last action performed by PADE related to hosts, files or commands. The Sort button is used to alphabetize the host list and the file list of each host.

5.8 The Graph Window

The Graph button in the PADE main windows pops up the Graph window (Figure 5.5), which is used to enter PADE data by graphical means. Alternatively, these data can be entered textually by using the Setup button (Section 5.5).

The main part of the window contains a graphical representation of the PADE data. Each host is represented by a circle with the hostname above it. The files to be transfered to each host are represented by rectangles attached to the circles and labeled with the name of the file on the development host.

Clicking on a circle brings up the host name and commands information in the top of the Graph window. Clicking on a rectangle brings up the information about a file. Double-clicking on a rectangle brings up the Edit File window, equivalent to the Edit File button in the PADE Files window (Section 5.6).

File designations can be moved from one host to another by dragging the rectangle with the left mouse button, and dropping it on the circle of the other host. The functionality of copying a file from one host to another is accomplished by first copying the file (see below) and then moving it to the other host.

For most operations in the menus under the Graph window buttons, you select the action from the menu, and then click on the circle or rectangle to which the action is to be applied.

The buttons in the Graph window are:

- Hosts – The operation of the menu under this button is identical to the menu under the Setup button of the PADE main window (Section 5.5).
- Files – The operation of the menu under this button is identical to the buttons in the PADE Files window (Section 5.6), except that when an action is selected you will be prompted (in the top of the Graph window) to click on the circle(s) or rectangle(s) to which the action applies.

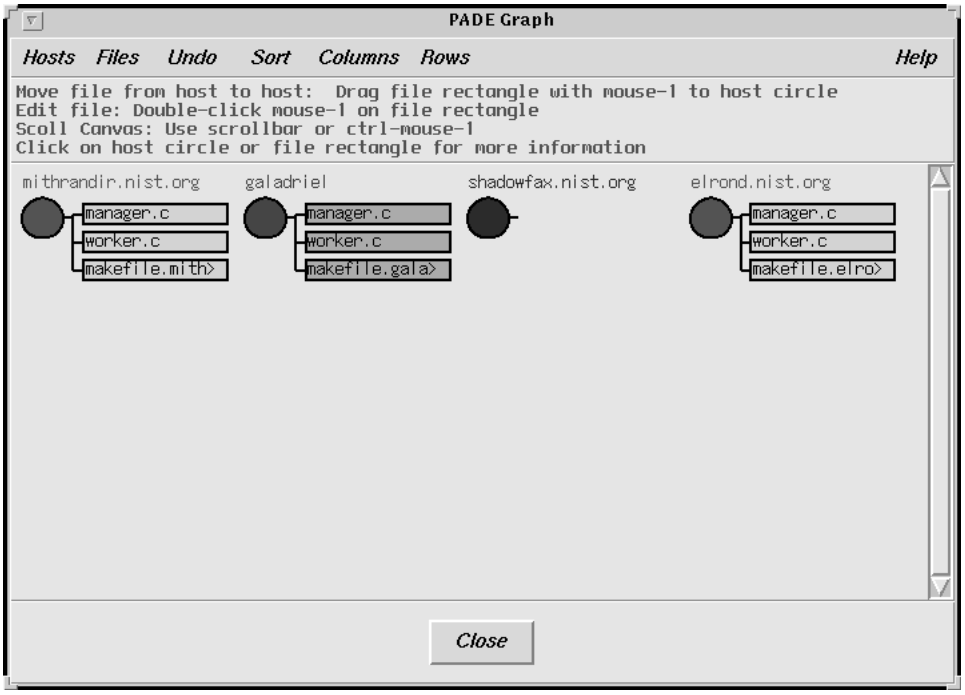

Figure 5.5: The Graph window

- `Undo` – identical to the `Undo` button in the PADE main window (Section 5.7).
- `Sort` – alphabetizes the hosts and the files within each host.
- `Columns` and `Rows` – these two buttons change the layout of the hosts in this window. `Columns` determines the number of hosts displayed in a horizontal line. `Rows` determines the number of files that will fit vertically in the window.
- `Help` – displays a summary of some of the operations in the `Graph` window near the top of the window.

5.9 The `PVM` Menu

The `PVM` menu is used to configure and execute PVM and related programs. It is pictured in Figure 5.6. The menu entries are:

- `PVM console` – brings up a PVM console, and starts PVM if it is not already running. If a PVM hostfile has not already been selected (see below), then the user will prompted to create or select a hostfile, or to start PVM without a hostfile. The PVM console will come up in a term window (for example *xterm*, but the type is determined by user preferences, Section 5.4), and may be left up or closed with the PVM console command *quit*. It is recommended that the console window be left open in case the PVM application "hangs" (which will also cause PADE to hang), in which case the console is required to terminate the application.
- `PVM daemon` – starts the PVM daemon without bringing up the console. As with `PVM console` above, the user will be prompted if a hostfile has not been selected.

Figure 5.6: The PVM menu

- `xpvm` – launches the *XPVM* application, a GUI interface to PVM.
- `pvmstat` – runs the CAML Utility *pvmstat*. This returns information about the status of the virtual machine.
- `pvmtidy` – runs the CAML Utility *pvmtidy*. This removes all spurious *pvmd.** files from the nodes of the virtual machine and kills any hopelessly lost PVM daemons, in case PVM was not shut down gracefully.
- `Pick Hostfile...` – selects an existing file as the PVM hostfile. It brings up a file browser.
- `Edit Hostfile...` – creates or edits a PVM hostfile. It brings up a `PADE Edit File` window.
- `Write Host list to Hostfile...` – creates a PVM hostfile. The file will contain the same list of hosts as in the current PADE data. It brings up a file browser to select the file to write the host list to.
- `List Hostfile` – displays the current PVM hostfile.

5.10 The `PVMmake` Menu

The `PVMmake` menu (Figure 5.7) is used to create the *PVMmake* config file (see Chapter 4) and run *PVMmake*.

The entries are:

- `Set Files and Commands to Save` – opens another window (Figure 5.8) in which you select a subset of hosts to send commands to, and, for each host, a subset of files to send. This gives PADE

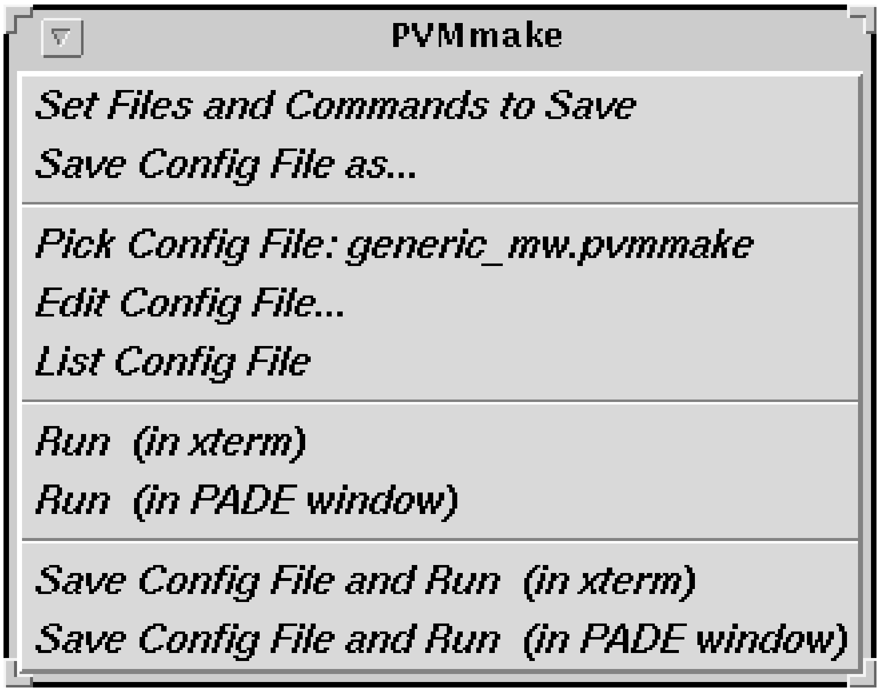

Figure 5.7: The PVMmake menu

nearly as much power for parallel applications as is found in the widely used *make* command for sequential applications in a Unix environment. The left half of the window allows you to choose which hosts will receive commands, and to toggle between hosts for the setting of which files to send. The right half of the window allows you to choose which files are to be sent to the selected host. The `Files modified since last PVMmake` button at the bottom will select those files that have been changed since the last time *PVMmake* was run. The version of this button on the left selects these files for all hosts, while the version on the right selects them for one host. If the last *PVMmake* run was with the current PADE data, then this acts like *make* in the sense that only those files that have been changed will be sent and recompiled. Be cautious, however, if *PVMmake* is being run with different *PVMmake* files, as in the "PVMmake clean" example in the tutorial (Section 3.2).

- `Save Config File as...` – writes a *PVMmake* config file containing the information in the PADE data, filtered by those items selected by `Set Files and Commands to Save`. It brings up a file browser to select the name of the config file. Be careful not to overwrite an unfiltered file with the filtered data.

- `Pick Config File: <current filename>` – selects an existing *PVMmake* config file. It brings up a file browser.

- `Edit Config File...` – creates or edits a *PVMmake* config file. It brings up a `PADE Edit File` window.

- `List Config File` – lists the current *PVMmake* config file in a scrollable window.

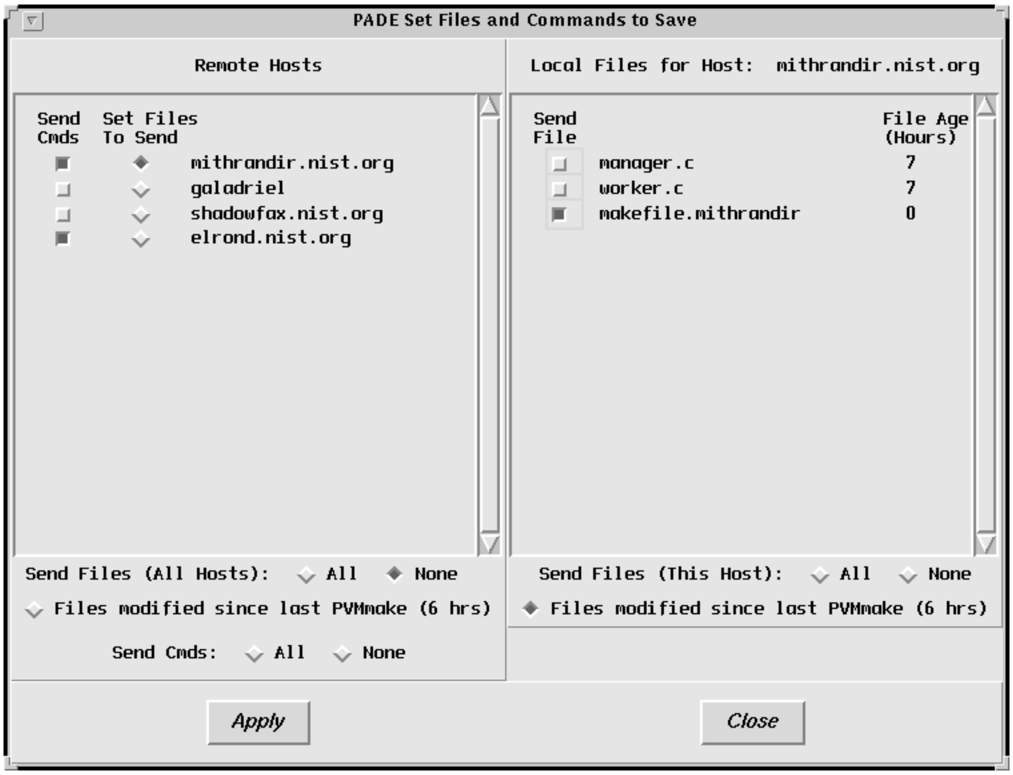

Figure 5.8: The Set Files and Commands to Save window

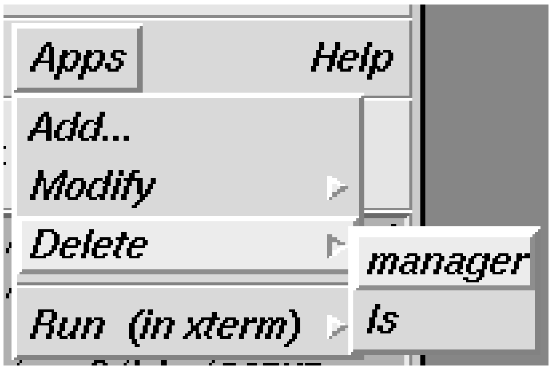

Figure 5.9: The Apps menu

- `Run (in xterm)` – runs *PVMmake* with the output going to an *xterm* window, or the type of term specified by the user's preference (Section 5.4).
- `Run (in PADE window)` – runs *PVMmake* with the output going to a scrollable window.
- `Save Config File and Run (in xterm)` – this is a shortcut that is equivalent to pressing `Save Config File as...`, selecting the file shown under `Pick Config File: <current filename>`, and pressing `Run (in xterm)`.
- `Save Config File and Run (in PADE window)` – same as previous button except output goes to a scrollable window.

5.11 The `Apps` Menu

The `Apps` menu (Figure 5.9) can used for running any command, for example PVM applications, performance monitoring software, Unix commands, etc. The most common use is to run the PVM application compiled by *PVMmake*. The commands are automatically saved in the *.paderc* file in the user's home directory and will appear in subsequent PADE sessions. For the `Modify`, `Delete`, and `Run` entries, the application or command is selected from a submenu of known commands.

The menu entries are:

- `Add...` – adds a new command to the list of known commands. It brings up a `PADE Entry` window for specifying the command or executable file.
- `Modify` – change the name of a command.
- `Delete` – remove a command from the list of known commands.
- `Run (in xterm)` – runs a command.

5.12 The `Help` Menu

The `Help` menu (Figure 5.10) contains several sources of assistance to the user. These come from World Wide Web pages, man pages and a system dependent help command. The entries in this menu are:

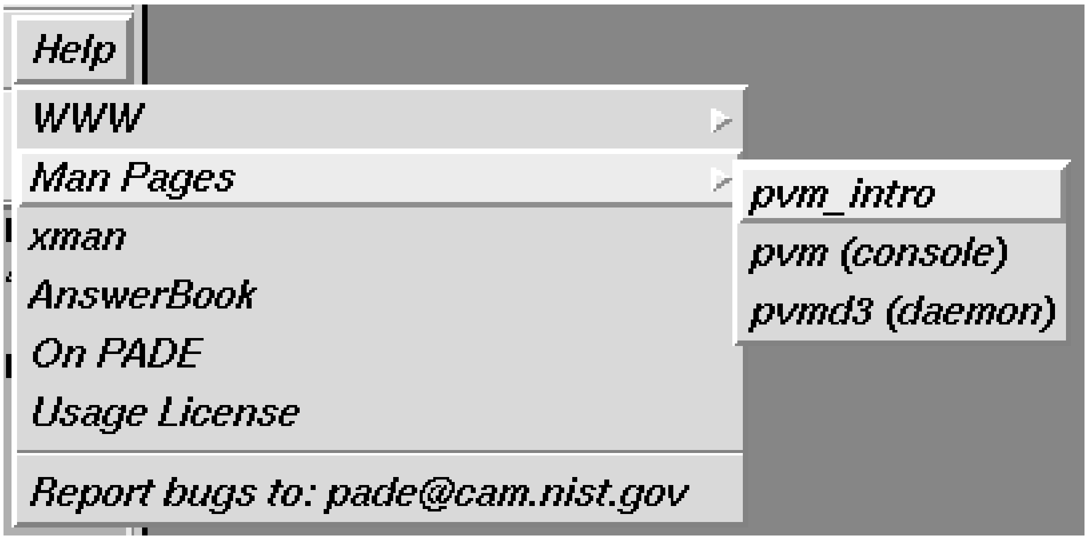

Figure 5.10: The Help menu

- `WWW` – selection of a submenu entry will spawn the WWW browser specified by the user preference (Section 5.4) and connect to the selected WWW page. The number and type of entries in this submenu will change over time except for the first two:
  - `PADE Home Page` – provides a link to the PADE home page at NIST.
  - `PADE Manual (local)` – provides a link to the local online PADE manual.
- `Man Pages` – selection of a submenu entry opens a scrollable window with the selected man page. Currently supports:
  - `pvm_intro` – general introduction to PVM and PVM environment variables
  - `pvm (console)` – description of the PVM console program.
  - `pvm3 (daemon)` – description of the PVM daemon program.
- `xman` – runs the X Window System interface to man pages.
- system dependent help command – this menu entry varies from system to system, and is some form of help command. Currently supported help commands are:
  - IRIS InSight on IRIX
  - InfoExplorer on AIX
  - Helpview on HP-UX
  - AnswerBook on SunOS

  Consult your local system documentation for information on running the above help commands.
- `On PADE` – prints a brief summary of the PADE operations in the bottom of the PADE main window.
- `Usage License` – prints the PADE usage license.