# **PowerWorks Linux Development Environment Tutorial**



0898100-000 April 2001

Copyright 2001 by Concurrent Computer Corporation. All rights reserved. This publication or any part thereof is intended for use with Concurrent Computer Corporation products by Concurrent Computer Corporation personnel, customers, and end–users. It may not be reproduced in any form without the written permission of the publisher.

The information contained in this document is believed to be correct at the time of publication. It is subject to change without notice. Concurrent Computer Corporation makes no warranties, expressed or implied, concerning the information contained in this document.

To report an error or comment on a specific portion of the manual, photocopy the page in question and mark the correction or comment on the copy. Mail the copy (and any additional comments) to Concurrent Computer Corporation, 2881 Gateway Drive, Pompano Beach, FL 33069-4324. Mark the envelope **"Attention: Publications Department."** This publication may not be reproduced for any other reason in any form without written permission of the publisher.

PowerWorks, PowerMAX OS, Power Hawk, NightBench, NightSim, NightTrace, NightView, and MAXAda are trademarks of Concurrent Computer Corporation.

Motorola is a registered trademark of Motorola, Inc.

Linux is a registered trademark of Linus Torvalds.

UNIX is a registered trademark of The Open Group.

Printed in U.S.A.

Revision History:	Level:	Effective With:
Original Release April 2001	000	PowerMAX OS 4.3

## **General Information**

The PowerWorks<sup>™</sup> Linux Development Environment (PLDE) allows users on a Linux<sup>®</sup> PC to develop applications for Concurrent real-time computer systems. The PLDE provides cross compilation, cross linking, and cross debugging and analysis tools. Editing, compilation, linking, and scheduling, as well as debug and analysis sessions, are hosted on the Linux system while the application programs execute on a system running Concurrent's PowerMAX OS<sup>™</sup> real-time UNIX<sup>®</sup>-based operating system.

The PowerWorks Linux Development Environment consists of high-performance Ada95 and C/C++ compilers, the NightView<sup>TM</sup> symbolic debugger, NightTrace<sup>TM</sup> event analyzer, NightSim<sup>TM</sup> frequency-based scheduler, and the NightBench<sup>TM</sup> GUI program development environment.

Utilizing the PLDE utilities on a Linux system while targeting the PowerMAX OS system offloads the heavy processing associated with compilation, linking, symbolic debug translation, and GUI network traffic from the real-time target systems.

## Scope of Manual

This manual is a tutorial for the PowerWorks Linux Development Environment.

## **Structure of Manual**

This manual consists of one chapter which is the tutorial for the PowerWorks Linux Development Environment.

## Syntax Notation

The following notation is used throughout this guide:

italic	Books, reference cards, and items that the user must specify appear in <i>italic</i> type. Special terms and comments in code may also appear in <i>italic</i> .
list bold	User input appears in <b>list bold</b> type and must be entered exactly as shown. Names of directories, files, commands, options and man page references also appear in <b>list bold</b> type.
list	Operating system and program output such as prompts and mes- sages and listings of files and programs appears in list type. Keywords also appear in list type.

## PowerWorks Linux Development Environment Tutorial

<u>emphasis</u>	Words or phrases that require extra emphasis use emphasis type.
window	Keyboard sequences and window features such as push buttons, radio buttons, menu items, labels, and titles appear in window type.
[ ]	Brackets enclose command options and arguments that are optional. You do not type the brackets if you choose to specify such option or arguments.
{ }	Braces enclose mutually exclusive choices separated by the pipe $( )$ character, where one choice must be selected. You do not type the braces or the pipe character with the choice.
	An ellipsis follows an item that can be repeated.
::=	This symbol means is defined as in Backus-Naur Form (BNF).

## **Referenced Publications**

The following publications are referenced in this document:

0890395	NightView User's Guide
0890398	NightTrace Manual
0890458	NightSim User's Guide
0890514	NightBench User's Guide
0890516	MAXAda Reference Manual

# Contents

## Chapter 1 Using the PLDE

Overview	1-1
Before you begin	1-1
Pathname conventions.	1-2
Remote shell access.	1-3
Privileges	1-3
Additions to PATH	1-4
Getting Started	1-5
Using NEdit	1-5
Using NightBench	1-8
Creating a new environment.	1-8
Introducing an existing source file into the environment	1-10
Creating a new source file in the environment	1-12
Setting compile options	1-15
Defining a partition.	1-16
Activating tracing for a partition	1-16
Building a partition.	1-17
Before you continue	1-18
Invoking NightSim	1-19
Using NightSim	1-20
Configuring the Scheduler	1-20
Scheduling a process	1-22
Activating user tracing and kernel tracing	1_22
Setting up the scheduler	1-25
Using NightView	1-20
Adding a tracepoint in the program	1 30
Adding a fracepoint in the program	1-30
Starting the simulation	1-32
Inserting a patchnoint	1-33
Histing a patenpoint	1-33
Disabling the natabasist	1-33
Disability the patchpoint.	1-33
Exiting the program	1-30
Kemoving ine scheduler	1-3/
Using Night Frace.	1-39
Converting kernel trace event files	1-39
Creating Night Trace configuration files	1-40
Invoking NightTrace.	1-41
Creating a default kernel page	1-42
Searching for a kernel trace event	1-43
Searching for a user trace event	1-46
Zooming in	1-47
Conclusion	1-49

## Illustrations

Figure 1-1. NEdit Editor	1-6
Figure 1-2. NightBench Project	1-8
Figure 1-3. Creating a new environment - language selection	1-9
Figure 1-4. Creating a new environment - specifications	1-9
Figure 1-5. Introducing an existing source file	1-11
Figure 1-6. Source file, <b>prog.a</b> - newly introduced	1-12
Figure 1-7. Creating a new source file	1-13
Figure 1-8. Setting environment-wide compile options	1-15
Figure 1-9. Activating tracing for a partition	1-17
Figure 1-10. Builder window - Build page for <b>prog</b> partition	1-18
Figure 1-11. Starting NightSim from NightBench	1-19
Figure 1-12. NightSim Scheduler	1-20
Figure 1-13. NightSim Edit Process	1-22
Figure 1-14. NightView Global Window	1-27
Figure 1-15. Start NightView Session on Remote Host dialog	1-28
Figure 1-16. NightView Dialogue	1-29
Figure 1-17. NightView Principal Debug Window	1-30
Figure 1-18. Setting a new tracepoint	1-31
Figure 1-19. Resuming execution	1-32
Figure 1-20. Starting the simulation	1-33
Figure 1-21. Setting a new patchpoint	1-34
Figure 1-22. Disabling a patchpoint	1-36
Figure 1-23. Resuming execution	1-37
Figure 1-24. Removing the scheduler	1-37
Figure 1-25. Removing the scheduler	1-38
Figure 1-26. NightTrace Main window	1-41
Figure 1-27. NightTrace display page	1-42
Figure 1-28. Default Kernel Page	1-43
Figure 1-29. Searching for a kernel trace event	1-44
Figure 1-30. First kernel trace event	1-45
Figure 1-31. NightTrace display page repositioned accordingly	1-45
Figure 1-32. Searching for a user trace event	1-46
Figure 1-33. NightTrace display page	1-47
Figure 1-34. Zoomed in kernel display page	1-48

# 1 Using the PLDE

Concurrent's PowerWorks<sup>®</sup> Linux Development Environment (PLDE) allows users on a Linux<sup>®</sup> PC to develop applications for any Concurrent real-time computer system. The PLDE makes it easy to utilize the features of Concurrent compilers and real-time GUI tools. Application programs are compiled and debugged directly on a Linux PC while targeted to a system running Concurrent's PowerMAX OS<sup>TM</sup> real-time UNIX-based operating system.

The PowerWorks Linux Development Environment consists of high-performance C/C++ and MAXAda<sup>TM</sup> (Ada95) compilers, the NightView<sup>TM</sup> symbolic debugger, NightTrace<sup>TM</sup> event analyzer, NightSim<sup>TM</sup> frequency-based scheduler, and the NightBench<sup>TM</sup> Program Development Environment.

## **Overview**

This is a demonstration of the PowerWorks Linux Development Environment. In this tutorial, we will use many of the PLDE tools including:

- NEdit
- NightBench
- MAXAda
- NightSim
- NightView
- NightTrace

integrating them together into one cohesive example.

Please see "Before you begin" on page 1-1 for some important recommendations and considerations.

## Before you begin

In order to run the portion of the tutorial that uses the NightSim Scheduler and the Night-View Source-Level Debugger, a system running PowerMAX OS should be networked to your Linux system. If you have a PowerMAX OS system networked to your Linux system, the following items must also be taken into consideration:

- Pathname conventions
- Remote shell access
- Privileges
- Additions to PATH

Proceed to "Getting Started" on page 1-5 to begin the tutorial.

#### NOTE

You may still run the tutorial (excluding the portions that use the NightSim Scheduler and the NightView Source-Level Debugger) even if you do not have a system running PowerMAX OS networked to your Linux system. You will be instructed as to how to skip over the sections that use the NightSim Scheduler and the NightView Source-Level Debugger.

## **Pathname conventions**

It is *highly recommended* that the paths from the host system (the system running Linux) and the target system (the system running PowerMAX OS) to the executables and working directories be identical.

Their mount points should be based on a common name.

Consider the following entries showing a Linux filesystem mounted via NFS on a Power-MAX OS system.

The following entry is located in /etc/fstab on the Linux system:

dev/hda5	/myspace	ext2	defaults	1	2

and the following entry is located in /etc/vfstab on the PowerMAX OS system

linuxsys:/myspace - /myspace nfs - yes rw,bg,soft

where linuxsys is the name of the host system running Linux.

In the above example, the user would create their working directory under /myspace. Creating a working directory is covered in "Getting Started" on page 1-5.

Ensure that your **umask** setting on the Linux system will allow the PowerMAX OS system to read and write files in your working directory, or use the same user and group ID on both systems. To automatically ensure that all files your user creates on the Linux system are publicly readable and writeable, include the following command in your shell startup script:

umask 000

See "Before you begin" on page 1-1 for other important recommendations and considerations.

## **Remote shell access**

Since NightSim uses **rsh** to start the *NightSim server* process on each *target system*, the user must be able to **rsh** to those systems.

Ensure that a login for your user name exists on the target system and

• the **.rhosts** file in the home directory for that user name on the target system contains an entry for your user name and the *NightSim host* 

An example entry might look like:

remote\_machine\_name username

where remote\_machine\_name is the name of the target system and username is your login name on the remote system.

Also note that the **.rhosts** file must have the permissions 644.

or

• the /etc/hosts.equiv file on the target system contains the name of the NightSim host

You may test your remote shell access by issuing the following command from your host system (the system running Linux):

/usr/bin/rsh remote\_machine\_name date

where **remote\_machine\_name** is the name of the target system. You should see the date and time on the remote system if successful.

See the **rsh(1)** man page for more details.

See "Before you begin" on page 1-1 for other important recommendations and considerations.

## **Privileges**

For the sections of the tutorial that run on the target system (the portions that use the NightSim Scheduler and the NightView Source-Level Debugger), this tutorial requires that the user have the following privileges on the target system:

- P\_CPUBIAS
- P\_PLOCK
- P\_RTIME

A convenient way to associate privileges with users is through the use of roles. A role is simply a named description of a set of privileges that have been registered for certain executable files, such as the shell. The system administrator creates roles and assigns users to

them. During the login process, users can request that their shell be granted the privileges associated with their role. Such a request takes the form of an invocation of the **tfad**-**min(1M)** command. Once privileges have been granted to the user's shell, subsequently spawned processes automatically inherit those privileges.

The following commands create a role and register all the privileges required by this tutorial to three commonly used shells (**sh**, **ksh**, and **csh**). The PowerMAX OS system administrator should issue the following commands once.

```
/usr/bin/adminrole -n PLDE_USERS
/usr/bin/adminrole -a sh:/usr/bin/sh:cpubias:plock:rtime PLDE_USERS
/usr/bin/adminrole -a ksh:/usr/bin/ksh:cpubias:plock:rtime PLDE_USERS
/usr/bin/adminrole -a csh:/usr/bin/csh:cpubias:plock:rtime PLDE_USERS
```

The following command assigns an example user (JoeUser) to the PLDE\_USERS role. The system administrator should issue the following command once.

/usr/bin/adminuser -n -o PLDE\_USERS JoeUser

JoeUser is now allowed to request that the above privileges be granted to his shell (assuming JoeUser utilizes either the sh, ksh, or csh shell, as these are the only shell commands registered in the PLDE\_USERS role). However, by default, these privileges are not granted. He must explicitly make the request by initiating a new shell with the tfad-min(1M) command. For convenience, it is recommended that the following command be added to the end of his .profile (or .login for csh users) file. (This file is executed during initialization of the login shell).

exec /sbin/tfadmin PLDE\_USERS: shell

where *shell* is the shell of your choice (**sh**, **ksh**, or **csh**).

See "Before you begin" on page 1-1 for other important recommendations and considerations.

## Additions to PATH

If users are interested in doing command-line compilations (although they are not covered in this tutorial), the following should be added to their PATH:

```
/usr/ada/bin
/usr/ccs/bin
```

See "Before you begin" on page 1-1 for other important recommendations and considerations.

## **Getting Started**

We will start by creating a directory in which we will do all our work. On the Linux system, create a directory and position yourself in it:

#### To create a working directory

- Use the **mkdir(1)** command to create a working directory. See "Pathname conventions" on page 1-2 for recommendations as to where you should create your working directory.

We will name our directory **tutorial** using the following command:

#### mkdir tutorial

- Position yourself in the newly created directory using the cd(1) command:

cd tutorial

## **Using NEdit**

Next, we will create one of the source files that will be used by our example program. We will do this using the NEdit Editor. NEdit is the PowerWorks Linux Development Environment editor. Although other editors may be used, NEdit comes with the PLDE and thus will be demonstrated in this tutorial.

Let's open the NEdit editor.

#### To start NEdit

- From the command line, type the following command:

neditor -ktalk &

#### NOTE

The **-ktalk** option allows NightBench to communicate with this NEdit session later in this tutorial. We specify the **&** so that this NEdit session runs in the background.

The NEdit Editor will be opened, ready to accept input.



Figure 1-1. NEdit Editor

We will enter one of the source files for our example program. This program is written in Ada and is shown below:

```
with data_process;
with rt_interface;
procedure prog is
    istat : integer;
    i : integer;
begin
    i := 0;
    rt_interface.FBS_wait( istat );
    while istat = 0 loop
        data_process.do_work;
        rt_interface.FBS_wait( istat );
        i := i + 1;
        end loop;
end;
```

This program utilizes the FBS\_wait service. FBS\_wait causes the calling process to go to sleep. The process will be awakened by a frequency-based scheduler at the process's scheduled frequency. At that point, it will enter the loop. The procedure data\_process.do\_work (which we will create later) will do some calculations. When do\_work returns from its processing, the program will encounter another FBS\_wait call which will cause the program to sleep until the frequency-based scheduler allows it continue.

### To save an untitled file using the NEdit Editor

- Select Save from the File menu. This will open a file dialog.
- Ensure the Directory is the same as the one you created in "Getting Started" on page 1-5.
- Enter the name prog.a in the Save File As field.
- Press OK.

Now that we have saved the file, we may close it in our NEdit session.

## To close a file in NEdit

- Select Close from the File menu.

We are finished using NEdit for this portion of the tutorial but will be using it in the next section with NightBench. We may leave this session open so that NightBench can communicate with it. However, if we chose to exit out of NEdit, NightBench would start its own session of NEdit, if necessary.

# **Using NightBench**

In order to compile and link our program, we will use the NightBench Program Development Environment. NightBench is a graphical user interface that provides a common work environment for the PowerWorks Linux Development Environment editor, compilers, and development tools. NightBench organizes all of the information required for consistent, repeatable development of PowerMAX OS applications while providing an efficient interface for editing, browsing, building, and debugging.

Let's open the NightBench Project window.

### To start NightBench

- From the command line, type the following command:

nbench

Note that we have not provided **nbench** with any parameters, indicating that we want to open the NightBench Project window. **nbench** accepts a number of command line options, allowing the user to open a particular NightBench component or to provide start-up information to NightBench.

The NightBench Project window will appear, listing the environments with which it has previously interacted. If no other environments have been created under NightBench, this list will be empty.

-			NightBench Project	
<u>N</u> ightB	ench	Options	Tools	Help
Enviro	nmen	ts		
Lang	Dire	ctory		
New	Ad	d to List	Remove from List Open Eulid	

Figure 1-2. NightBench Project

## Creating a new environment

One of the first steps we must take in order to use NightBench for program development is to create an *environment*. Environments are used as the basic structure of organization within NightBench.

#### To create a new environment from the NightBench Project window

- On the NightBench Project window, press the button marked New... so we can create our new environment. This will open a dialog in which you may select the language to be used in this environment.
- Select the language that will be used in this environment (Ada).
- Press the Next> button.

New	Environment		
Language for new environ	ment:		
Ada			
⊖ C/C++			
	Next>	Cancel	Help

### Figure 1-3. Creating a new environment - language selection

The next dialog presented allows us to specify details about the directory which will contain the new environment, the release of the compiler to be used, as well as the architecture of the target machine and the version of PowerMAX OS running on it:

Ne	w Environment	
Directory for new environm	nent:	
[/tooth/tutorial		
Compiler Release	Target Architecture	PowerMAX OS Version
phase3.3.1 (default) 💻	moto 🗖	4.3 🗖
☐ Create new environme	nt from script:	
Y		
<back< td=""><td>Done</td><td>Cancel Help</td></back<>	Done	Cancel Help

#### Figure 1-4. Creating a new environment - specifications

- Type the directory name in the Directory for new environment field where you want NightBench to create the new environment. This can be

the name of an existing directory or NightBench can create the directory for you. (Note that NightBench can only create a subdirectory of an existing directory.) We will enter the name of the directory we created in "Getting Started" on page 1-5. The full directory name in our example is /tooth/tutorial. Since we invoked NightBench from that directory, the pathname will appear in the Directory for new environment field.

- Select a **Compiler Release** if you have more than one release of MAX-Ada installed on your system. If you have only one release of MAXAda installed on your system, it will appear here.
- Choose a Target Architecture. Because we are building an executable that will run on a Concurrent real-time computer system, we must choose which type of system we are targeting. For our example, we will be targeting a Power Hawk<sup>TM</sup> 640 so we will select moto from the drop-down list. For more information on target architectures, see the section titled "Target Architectures" in the *MAXAda Reference Manual* (0890516).
- Identify the PowerMAX OS Version. In our example, we will select 4.3 from the drop-down list for the version of the operating system running on the system we are targeting.
- Press Done

This will add the new environment to the list of Environments in the NightBench Project window. NightBench will also open the new environment in its own NightBench Development window.

## Introducing an existing source file into the environment

Our next step is to populate the environment with *units*. Units are the basic building blocks for programs in NightBench. They are contained within source files and it is through these source files that they are introduced into their intended environments.

Source files may already have been created outside the NightBench environment or you may use the editing features of NightBench to create a new file. For this example, we will introduce the file, **prog.a**, that we created earlier in "Using NEdit" on page 1-5.

### To introduce an existing source file into a NightBench environment

- Click on the Source Files tab of the NightBench Development window.
- Press the Introduce/Create... button. This will open the Introduce Source Files dialog so we can introduce our source files (and the units contained within) into the new environment.

F		Introduct Source Falses	E S
Directory (tooth/tutorial)	1	Files to Introduce Pile Narw	Preprocess Create
Fiter	Uptime	Page .	
Exclude files     Show files it     Foliow symb     Directories     -	Place and an and a second seco		
Selection			E CONTRACTOR
OK	And All Create	Compet	Help

### Figure 1-5. Introducing an existing source file

- Maneuver to the directory in which the **prog.a** source file is contained. You may type the path to the directory name in the **Directory** field or use the entries in the **Directories** list to navigate to the desired directory. Since we invoked NightBench from the directory in which our file resides, we should already be positioned in that directory.
- Select the **prog.a** source file by clicking once with the mouse on the name in the Files list. The name of the source file will then appear in the Selection field.
- Press the Add button to add this file to the list of Files to Introduce. (You may introduce any number of files, or create new files, by adding them here but for our example we will just introduce this one file.)
- Press the OK button to introduce the source file into the environment.

The unit prog that was contained in the source file **prog.a** is now a part of the environment /tooth/tutorial.

The source file now appears in the list of files on the Source Files page of the Night-Bench Development window and the unit contained within now appears on the Units page.

Dependencies	Partitions
Dependencies	Partitions
Dependencies	Partitions
Last Modified	_ Full Pathname
Last Modified	
	Units Preproce
1171500 12-44 13	1 no

Figure 1-6. Source file, prog.a - newly introduced

## Creating a new source file in the environment

As mentioned earlier, source files may already have been created outside the NightBench environment or you may use the editing features of NightBench to create a new file. In this part of the tutorial, we will create a new source file using the editing features of NightBench.

### To create a new source file in a NightBench environment

- Click on the Source Files tab of the NightBench Development window.
- Press the Introduce/Create... button. This will open the Introduce Source Files dialog so we can create new source files and introduce them into the new environment.

	Introduce Source Files	1412
Directory	Files to Introduce	International Provide
(tooth)utorial/	The Horw	
Fiter		
[*ada *.a *.pp *.adb *.ads		
Exclude files already introduced.     Show files in subdirectories.     Ender subdirectories.		
Directories Files		
Selection		
data_process.a	1	12
Add Add Add Create	Commands # Add files to list using short mane.	Parallel Intra's [1 4 -
	Cantel	Help

Figure 1-7. Creating a new source file

- Maneuver to the directory in which you would like to create the new source file. You may type the path to the directory name in the Directory field or use the entries in the Directories list to navigate to the desired directory. Since we invoked NightBench from the directory in which we would like to create our new source file, we should already be positioned in that directory.
- Enter the name of new source file in the Selection field. For our example, we will name our file data\_process.a
- Press the Create button. This will add the file name to the list of of Files to Introduce. Note the Yes in the Create column for this file. (You may create other new source files, or introduce other existing source files, by adding them here but for our example we will just create this one file.)
- Press OK.

For each file with a Yes in the Create column of the list of Files to Introduce, an editor window will be opened. This will bring up the editor that NightBench is configured to use. NEdit is the default editor for NightBench. See the section titled "Preferences - Editor" in the *NightBench User's Guide* (0890514).

We may now enter the other source file used in our example program:

```
package data_process is
   iteration_count : integer := 1;
   x1,x2 : long_float;
   x1_mult_x2 : long_float;
   procedure do_work;
end data_process;
package body data_process is
   procedure do_work is
   begin
     x1 := 1.0e-160;
     x2 := 1.0e-160;
      iteration_count := iteration_count + 1 ;
      for i in 1 .. 2000 loop
        x1_mult_x2 := x1 * x2 ;
      end loop;
   end;
```

```
end data_process;
```

#### To save a named file using the NEdit Editor

- Select Save from the File menu. Since we specified the pathname of the file to NightBench, NEdit saves our input in that file.

Now that we have saved the file, we may close it in our NEdit session.

## To close a file in NEdit

- Select Close from the File menu.

## Setting compile options

In order to debug the program using the NightView Source Level Debugger, we need to compile the program with debug information. We do this by setting an environment-wide compile option which will apply to all units within the current environment.

#### To set environment-wide compile options

- Click on the Settings tab of the NightBench Development window.
- Press the Show Options Editor button associated with the Permanent Compile Options.
- On the General page of the Ada Environment Compile Options dialog, select full (2) from the drop-down list under the Permanent column for Debug Information.
- Press OK.

option	Default	Permanent	Effective
ebug Information	none (0)	full (2)	full (2)
ptimization Level	minimal (1)	(unspecified) -	minimal (1)
hare Mode	non-shared	(unspecified) -	non-shared
uppress Runtime Checks		4	D
Compiler Error Output	errors only	(unspecified) -	errors only
luiet Compiler Info Msgs.		4	
luiet Compiler Warnings		*	

Figure 1-8. Setting environment-wide compile options

## NOTE

Alternatively, you could have entered **-g** in the **Permanent** Compile Options field on the Settings page and pressed the Apply button.

## **Defining a partition**

In order to use the units introduced into NightBench, we must include them in a *partition*. NightBench defines three types of Ada partitions:

- active
- archive
- shared object

For our example, we want to include our prog unit in an executable program so we will be defining an *active* partition.

#### To define all active partitions in the environment

- Click on the Partitions tab of the NightBench Development window to get to the Partitions page.
- Press the Create All button. This creates an active partition for each unit in the current environment that qualifies as a main unit.

## Activating tracing for a partition

We will need to activate tracing for this partition so that we may generate trace data when we run the program and then subsequently analyze it using the NightTrace Analyzer.

#### To activate tracing for a partition

- Click on the Partitions tab of the NightBench Development window to get to the Partitions page.
- Click on the Tracing settings tab.
- Check the Activated checkbox.
- Select the Mechanism to be used for tracing. In our example, we will select ntraceud from the drop-down list so that the Ada run-time executive can log trace events using the NightTrace user daemon, ntraceud.
- Press Apply.

Qevelopment       Select       Options       Tools       Her         Ada Environment       Acoth/tutorial				RightDenth	Development.		
Ada Environment       Approximation         Settings       Source Files       Units       Dependencies       Partitions         Environment Filter       Local       Image: Source Files       Image: Source Files       Provisionment       Partitions         Reme       Kaid       Output File       Home Environment       Rue Context       prog         Image: Source All       Remove       Build       Pun Context       Pun       Debug       File         Create       Create All       Remove       Build       Pun Context       Pun       Debug       File         Settings for partition prog (active)       Image: Create All       Remove       Build       Pun Context       Pun       Debug       File         Settings for partition prog (active)       Image: Create All       Remove       Build       Pun Context       Pun       Debug       File         Settings for partition prog (active)       Image: Create All       Remove       Build       Pun Context       Pun       Debug       File         Settings for partition prog (active)       Image: Create All       Remove       Image: Create All       Remove       Expert         Tracing       Image: Activated       Image: Create All       Image: Create All       Image: Create All <th>Gevelopment</th> <th>Seject Opt</th> <th>ans Taab</th> <th></th> <th></th> <th></th> <th>H</th>	Gevelopment	Seject Opt	ans Taab				H
Environment Filter Local Herne Kind Output File Home Environment Rue Context prog setw prog prog Create Create AI Remove Build Pun Context Pun Debug File Settings for partition prog (active) General Units Link Control Link Options Tracing Expert Tracing ₹ Activated Event Logging ₹ Enabled (initial setting) Mechanism mtraceud = hstrumentation ₹ Runtime Events Turing Source deficit = ↓ Library Unit Elaboration Override Buffer Size	Ada Environne Settings	Int /tooth/tute	rial Eiles	Units	Dependenc	ies <u>P</u> artitie	ons
Name       Kind       Output File       Home Environment       Run Context         prog       active       prog       prog       prog         Create       Create AI       Remove       Build       Pun Context       Pun       Debug       File         Settings for partition prog (active)	Environment Fi	ter Local	-				
reage where prog prog prog prog prog prog prog prog	liame	Kind (	Sutput File	Home Env	ironment	Run Contesct	
Create       Al       Remove       Build       Run Contest       Run       Debug       File         Settings for partition prog (active)       □       Rew Link Option       □       Rew Link Option         General       Units       Link Control       Link Options       Tracing       Expert         Tracing       ✓ Activated             Event Logging       ✓ Enabled (initial setting)       Mechanism       mtraceud =          Instrumentation       ✓ Runtime Events       Thring Source       default          ✓ Library Unit Elaboration       Override Buffer Size	arog	active p	rog			head	
Create       Al       Remove       Build       Pun Context       Pun       Debug       File         Settings for partition prog (active)       □       Rew Link Option       □       Rew Link Option         General       Units       Link Control       Link Options       Tracing       Expert         Tracing       ✓ Activated              Event Logging       ✓ Enabled (Initial setting)       Mechanism       mtraceud =          Instrumentation       ✓ Runtime Events       Timing Sourch       default          ✓ Library Unit Elaboration       Override Buffer Size							
General       Units       Link Control       Link Options       Tracing       Eggent         Tracing       ✓ Activated         Event Logging       ✓ Enabled (Initial setting)       Mechanism       mtraceud →         Instrumentation       ✓ Runtime Events       Tiring Source       default         ✓ Library Unit Elaboration       Override Buffer Size       ✓	Create	reate Al	Remove	Build	Pun Context	Run Debug	File
General     Units     Link Control     Link Options     Tracing     Expert       Tracing     ✓ Activated           Event Logging     ✓ Enabled (initial setting)     Mechanism     mtraceud →       Instrumentation     ✓ Runtime Events     Tiring Source     defest       ✓ Library Unit Elaboration     Override Buffer Size	Settings for pa	ntition prog (a	ctive)				- Rew Link Optio
Tracing     ✓ Activated       Event Logging     ✓ Enabled (Initial setting)     Mechanism     mtraceud →       Instrumentation     ✓ Runtime Events     Thring Source     default       ✓ Library Unit Elaboration     Override Buffer Size     ✓	General	Units	Lin	Control	Link Options	Tracing	Egert
Event Logging V Enabled (Initial setting) Mechanism miraceud at Instrumentation V Runtime Events Thiring Source: defect at V Library Unit Elaboration Override Buffer Size	Trading	Activate	d				
Instrumentation IV Runtime Events Turing Source detect -	Event Logging	Enabled	i (Initial setti	ng)	Mechanism	mtrace	ud
V Library Unit Elaboration Override Buffer Size	Instrumentatio	n 🗑 Runtime	Events		Thring Sou	tar delas	
A CORA CALCINOCATION COLUMN COLUMN 2558		W Linear	Init Elabory	600	Override Pu	Mar She 1	in the second se
		in reality	CULCHOON	en la compañía de la	Overnoe bu	nei ora	
	-						
	Apply Cave	Environe	ment Link O	ptions			

Figure 1-9. Activating tracing for a partition

## **Building a partition**

At this point, we have an environment, /tooth/tutorial, that has within it the definition for the active partition, **prog**, made up of a main unit, **prog**, contained in the source file, **prog.a**, and another unit data\_process, contained in the source file, **data\_process.a**. Full debug information will be generated for the program and tracing has been activated so that we may gather tracing data for later analysis.

We can now build this partition. We do this using the NightBench Builder.

### To build the partition

- In the list of partitions on the Partitions page, make sure the partition **prog** is selected.
- Press the button marked Build. This will open the NightBench Builder window so we can build our new partition.

In Figure 1-10, you will see that partition prog has been automatically entered in the Targets field on the Build page. This is because it was selected on the Partitions page when the Build button was pressed.

- Press Start Build.

The Build Progress bar shows the number of actions (compilations and links) left to perform in the current build as the Transcript window details each step taken during the build.

Tarrato	and time non	arrs   Tr	meida	Testice	Run	Expres	
rargets	Start Rule	Vertu Billet	Sion	c	antext	Bun	Detroi
Transit		Tanay Grand	Company	-		1941	C CONT
compi compi compi linkin Build co	ling: package ling: package ling: subprogr ng: prog mpleted Hon No	spec data_proce body data_proce an body prog v 20 15:05:23 2	300				
coopi coopi linki Julid oo	Ling: package Ling: package Ling: subprogr mpleted from No evidus We	opec data_proce body data_proce as body prog v 20 16:06:23 20 w Niccii Viewi	ng transcr	ipt 4 of 4			

Figure 1-10. Builder window - Build page for prog partition

When the build is completed, a Build Completed dialog notifies the user.

### NOTE

The notification operations can be changed on the Notification page.

## Before you continue

The following sections require a PowerMAX OS system networked to your Linux system since those sections use the NightSim Scheduler and the NightView Source-Level Debugger (see "Before you begin" on page 1-1 for important recommendations and considerations concerning this configuration).

However, if you do not have a PowerMAX OS system networked to your Linux system, you may jump to the section "Using NightTrace" on page 1-39 and continue with the tutorial.

## Invoking NightSim

Because this program uses the frequency-based scheduler, we will use the NightSim Scheduler to schedule the process.

NightBench allows the user to invoke the NightSim Scheduler directly.

### To invoke the NightSim Scheduler from the NightBench Program Development Environment

- Select NightSim Scheduler from the Tools menu of either NightBench Development or the NightBench Builder.

Options	Tools	
/tutorial	NightBench Project	
ce <u>F</u> iles	NightBench Development	endencies
	Night <u>B</u> ench Builder	
al .	NightView <u>D</u> ebugger	
Output	NightSim Scheduler	Run Co
pr <b>og</b>	View File	prog
	 <u>E</u> dit File	
Rem	nove Build Run Co	ntext Run

Figure 1-11. Starting NightSim from NightBench

# Using NightSim

NightSim is a tool for scheduling and monitoring real-time applications which require predictable, repetitive process execution. NightSim provides a graphical interface to the PowerMAX OS frequency-based scheduler and performance monitor. With NightSim, application builders can control and dynamically adjust the periodic execution of multiple coordinated processes, their priorities, and their CPU assignments. NightSim's performance monitor tracks the CPU utilization of individual processes and provides a customizable display of period times, minimums, maximums, and frame overruns. For more information on NightSim, refer to the *NightSim User's Guide* (0890480).

# **Configuring the Scheduler**

The NightSim Scheduler window is opened, ready for us to configure it for our particular simulation.

jeh.		Nettin St	tedsler			11/15
NightSin Scheduler						Help
NightSim Hast: raphar						On-Line
Configuration file: (unnamed)						
Scheduler key	Timing host		Scheduler	Simulation	F	un Status
Cycles per frame	Clistribution:	None -	-	neis 💦	Frane:	+
Max. tasks per cycle:	Timing source: End	of processing cycle -	B Benn-e	910	Cycle:	হয়
Max tasks in scheduler:	O took gestant	piec	leader	Stopped Fidsume	On Target	No scheduler 斗
Permissions 00	0 m (min	10. www-695090)			Relat	1.0 sec
Target Supton FPID	Program Name	CPU Blas Pices	Prio- Sofe rite Paran Lin.	Halt/ Start Ci	cle Evecut	ion Schedule
						4
Edt. Dente Fite	Hetaih			12	-	P

Figure 1-12. NightSim Scheduler

### To configure a NightSim Scheduler

- Specify a Scheduler key. The key is a user-chosen numeric identifier with which the scheduler will be associated. For our example, we will use 100.
- Specify the Cycles per frame. This field allows you to specify the number of cycles that compose a frame on the specified scheduler. We will use the value 1.
- Specify the Max. tasks per cycle. This field allows you to specify the maximum number of processes that can be scheduled to execute during one cycle. Enter 10 for our example.

- Specify the Max. tasks in scheduler. This field allows you to specify the maximum number of processes that can be scheduled on the specified scheduler at one time. For our example, we will specify the value 10.
- Enter the name of a PowerMAX OS system which will act as the Timing host for the simulation. You may use the drop down list associated with this field for the names of systems previously used as timing hosts. For ourexample, we will enter buzzard, a Power Hawk 640 system.

#### NOTE

When NightSim is operating in On-Line mode, an attempt will be made to communicate with the system specified as the timing host. The user may experience a slight delay and the message Talking to Server... will appear in the Configuration File Name Area of the NightSim Scheduler as this occurs. See the *NightSim User's Guide* (0890480) for more information.

- Select a Timing source from the list provided. This list contains the set of devices available on the timing host. We will use Real-time clock 0c2.

#### NOTE

Do not use Real-time clock 0c0 for the Timing source as it is typically used by system utilities and could cause unwanted effects if used. See hrtconfig(1) for more information

Since we are using the real-time clock on the target system, we need to specify the clock period. For our simulation, we would like the real-time clock to "fire" every .5 seconds (or 500 milliseconds).

#### IMPORTANT

The following steps should be performed in the order presented below to ensure the correct value for the clock period.

- Choose the msec from the drop-down list next to the Clock period field.
- Specify Clock period. For our example, we will specify 500 for the number of milliseconds.

## Scheduling a process

Once we have properly configured the Scheduler, we can add a process to the frequency-based scheduler.

FPID	(No process selected)	
Target System:	buzzard Z	On-Line
Process Name:	Nooth/tuterial/prog	Select.
Working Directory:	/hooth/tutorial/	Select.
Debug:	F Schedule program within a NightView dialogue	
Terminal	□ Schedule program within an >term window	
Read Input From:	File Standard Input Stream	
input File:	Joesma	Select.
Send Output To:	🔄 File 🐞 Standard Output Stream	
Output File:	Inernal	Seleci.
Privleges:	cpubias, decread, dev, owner, plock, rtime, setspriv, setupriv, shmbind,	Select.
CPU Blas:	01234587831011 FAICPUs	
Sched. Policy:	FIFO Round Robin Other (Interactive)	
Priority:	50 (0-lowest, 59-highest)	
Parameter:	[ (optional)	
Salt Ovm, Limit:		
Overrun:	Halt FBS on frame overrun	
Starting Cycle:	(D-lawest, O-highest)	
Period	1 (0-unscheduled, 1-lowest, 1-highest)	

### Figure 1-13. NightSim Edit Process

### To add a process to the frequency-based scheduler

- Press the Edit... button on the NightSim Scheduler window. This will bring up the Edit Process window.
- Press the Select... button next to the Process Name field. This brings up the Select a Program dialog.
  - Since NightSim was invoked from our NightBench environment, the Directory field should coincide with our working directory. If it does not, either type the full pathname to our working directory,

/tooth/tutorial, in the Directory field, or maneuver to that directory using the items in the Directories list.

- Choose the program we wish to schedule from the Files list. For our example, we will select prog from the list.
- Press OK to select the program.
- Ensure that the Working Directory is the same directory that contains our program (the directory of the Process Name selected in the previous step).
- Check the Schedule program within a NightView dialogue checkbox. This will bring the program up in the NightView debugger before the program executes, allowing us to set *tracepoints* so that we may generate trace data when the program executes.
- Specify the Priority for this process. The range of priority values that you can enter is governed by the scheduling policy specified. NightSim displays the range of priority values that you can enter next to the Priority field. Higher numerical values correspond to more favorable scheduling priorities. For our example, we will give the process a priority of 50.
- Select Starting Cycle. This field allows you to specify the first minor cycle in which the specified program is to be wakened in each major frame. We will choose the lowest value, 0, for our example.
- Select **Period**. This field allows you to establish the frequency with which the specified program is to be wakened in each major frame. Enter the number of minor cycles representing the frequency with which you wish the program to be wakened. For our example, we will specify a period of 1, indicating that the specified program is to be wakened every minor cycle.
- Press Add to add the process to the frequency-based scheduler.
- Press the Close button to dismiss the Edit Process window.

## Activating user tracing and kernel tracing

At this point in the tutorial, we are about to create the scheduler configured according to the parameters we just specified and allow the program to run. However, we would like to generate trace data from this program while it is running so we need to start the Night-Trace user daemon (which we specified in the section titled "Activating tracing for a partition" on page 1-16) to log user trace events as well as KernelTrace which will collect data about the execution time of interrupts, exceptions, system calls, context switches, and I/O to various devices.

### To activate the NightTrace user daemon

- Log into the PowerMAX OS system where you will be running your simulation. This is the system we specified as the Timing host in the NightSim Scheduler window (see "Configuring the Scheduler" on page 1-20). So, for our example, we will log into the system **buzzard**.

 Position yourself in the working directory you created in "Getting Started" on page 1-5. Also, see "Pathname conventions" on page 1-2 for recommendations as to where you should create your working directory.

#### IMPORTANT

It is essential that you are positioned in the working directory that is associated with the user program being scheduled with NightSim. The NightTrace user daemon will communicate with the user program based on the file argument supplied in the next step.

- Invoke the NightTrace user daemon. We issue the **ntraceud** command which takes as an argument the name of a file in which to save the trace data. This file should be named *program\_name.trace.data*, where *program\_name* is the name of the program generating the trace data.

#### NOTE

By default, **ntraceud** requires write access to system SPL devices, e.g. /**dev/spl**, /**dev/spl1**, etc. On most systems, these devices are only writeable by the root user; therefore, you should run the **ntraceud** command as root.

However, since the use of SPL devices is not strictly necessary for tracing single-threaded user applications (although, for optimal real-time performance it is recommended), the **-ipldisable** option to **ntraceud** is acceptable.

Since the application in this tutorial is single-threaded, you may use the **-ipldisable** option as indicated below.

For our example, we will issue the following command:

```
ntraceud -ipldisable prog.trace.data
```

Now we can activate kernel tracing.

#### To activate kernel tracing

- Log into the PowerMAX OS system where you will be running your simulation. This is the system we specified as the Timing host in the NightSim Scheduler window (see "Configuring the Scheduler" on page 1-20). So, for our example, we will log into the system **buzzard**.
- Position yourself in a directory local to the PowerMAX OS system.

- Invoke the KernelTrace utility. We issue the **ktrace** command which can take a number of arguments.

#### NOTE

The KernelTrace utility requires root access in order to run. Ensure that the Linux system has exported its filesystem to the PowerMAX OS system in a manner which allows root to write on that file system. This normally requires the root\_no\_squash option in the /etc/exports entry for the file system. For example:

/myspace pmax\_system(rw,root\_no\_squash)

Alternatively, you can run the KernelTrace utility on a file system local to the PowerMAX OS system and subsequently copy its output file (as a non-root user) back to Linux file system. The remainder of the steps below assume that root has write access to the Linux file system.

We will use the -o option which specifies the name of a file in which to save the kernel trace data.

When generating kernel trace data, the resultant file can grow extremely large very quickly. In order to circumvent any problems that may arise from the output file growing extremely large, we will use the **-bufferwrap** option which limits the size of the output file. Specifying a value of 50 to this option will limit the size of the resulting output file to a little over 2 megabytes.

#### NOTE

Due to a problem with the **-bufferwrap** option, user and kernel data may not appear synchronized when viewing the trace data in subsequent steps. This problem has been fixed in the **ktrace** and **ntfilter** commands in PowerMAX OS 4.3 Patch Set 6 (**trace-004** and **base-006**). If these packages are not installed on your system, you may omit the **-bufferwrap** option. However, be aware that the kernel trace file may grow extremely large in a short period of time.

So, for our example, we will issue the following command, as the root user:

#### ktrace -bufferwrap 50 -o prog.ktrace.data

You should see output similar to the following:

locking into memory
setting priority to RT 59
open /dev/trace
initialize

set trace event time stamp source to Motorola Time Base Register gather trace point data

## Setting up the scheduler

### To set up the scheduler

- In the NightSim Scheduler window, press the Set up button.

This action:

- creates a scheduler that is configured according to the parameters we specified
- schedules the processes that we have added to the NightSim Scheduler window and starts them running up to the first FBS\_wait call, and
- attaches the timing source to the scheduler.

Because we have specified the Schedule program within a NightView dialogue option when we added this process to the frequency-based scheduler (see "To add a process to the frequency-based scheduler" on page 1-22), the NightView Source Level Debugger will be started.

## **Using NightView**

NightView is a graphical source-level debugging and monitoring tool specifically designed for real-time applications. NightView can monitor, debug, and patch multiple real-time processes running on multiple processors with minimal intrusion. In addition to standard debugging capabilities, NightView supports application-speed eventpoint conditions, hot patches, synchronized data monitoring, exception handling and loadable modules.

Because we have specified the Schedule program within a NightView dialogue option when we added this process to the frequency-based scheduler (see "To add a process to the frequency-based scheduler" on page 1-22), we are presented with a NightView Global Window as well as a dialog allowing us to log into the target system on which we will be running our program.



Figure 1-14. NightView Global Window

Remate host	buzzent	
Login name	usemand	
Password	I	=
Name for new Dialogue	bezard	_
Priority	1	- -
Nice Value	I	<b>N</b>
Scheduling Class	Time Sharing	-
Time Quantum	I	
Binding Type	System Default	-
CPU		2 113 114 115
	D1 D7 D	D 9 1D 11
NUMA.	Default Syste	m Default 🛁
	Text Def	aut
	Private Def	eut -
	Shared Def	aut -
	UBlock Def	aut 🔤

#### Figure 1-15. Start NightView Session on Remote Host dialog

### To start a NightView session on a remote host

- In the Start NightView Session on Remote Host dialog, ensure that the values for Remote host and Login name are correct.

### NOTE

The Name for new Dialogue field is initialized to the name of the remote system on which we are debugging our program.

- Enter your Password for the Login name on the system listed in the Remote host field.
- Press OK.

When the login has completed successfully, a NightView Dialogue window for the remote host will be opened as well as a Principal Debug Window with the execution of the program stopped.

el	NightView	Italogue: butterd	14
NightView	Dialogue		Help
buzzard		M	lachine: buzzard
Messages:			
IE-SI ashProcH (envio=13	ethod-0121 ) Permission denied		
Dislogue I/O:	Run your programs in	this shell.	
buzzar@ od / buzzar@ -bF nzis.*urtep: fpid 8 assign	toeth/tutorial/ -y50 -w0 -f1 -onohalt Begin ed ta process /tooth/tu	-wtooth/tutorial/prog torial/prog	د. 12
Qualifier:	Command:		Interrupt
buzzard	I		2
	Process	es for this Dialogue	
	PID:	Program name:	
	22776	PL08	_
	Detach	КЛ	

## Figure 1-16. NightView Dialogue

During initialization, you will see a message similar to the following:

```
Warning: Process buzzard:3336 is no longer debuggable,
detaching.
[E-SlashProcMethod-012]
    (errno=13) Permission denied
```

This is an anomaly caused by an intermediate process which schedules the user program. You may ignore this warning.

-		Sight	lies Principal	Bebug Window		
NightView	Process	Source	Eventpoint	Mew		Help
Messages						
Reading by Executable /tooth/but Switched to ]	mbols from /ts file set to orial/prog o process bugg	eth/tutorsa ard:22776.	L/progdone			
prog						buzzardt22776
f prog	a				St	opped for exer
4 • 1 pro 5 1 6 1 7 1 beg 9 • 1 10 1 112 1 12 1 13 • 1 15 • 1 15 • 1 15 • 1 15 • 1 15 • 1 17 • 1 18 1 19 9 10 1 10 1 11 2 11 2 11 2 11 2 12 1 13 1 14 1 15 • 1 15 • 1 15 • 1 16 1 17 1 18 1 19 1 19 1 10 10 1 10 10 1 10 10 1 10 1 10 10 1 10 1 10 1 10 10 10 1 10 10 10 10 10	cedure prog in initial : integer: in i : integer: in t := 0; rt_interface.f data_stuFF. rt_interfac i := 1 + 1; end loop;	r: PS_wait( in 0 loop do_work; e_FPS_wait(	tat ); istat );			
Flesume	Step Break	Next	Stepi	Nexti	Finish Clear	Stop
Gualfier: buzzard:2	Comr 2776 I	rand				_interrupt
Dialogue:	Gr :PID: E	oup of Pro Exec File	cesses for this : State: PEADER	Window		Switch To Stopped Process
		1	Seituh To			AUC

Figure 1-17. NightView Principal Debug Window

# Adding a tracepoint in the program

Since we would like to generate user trace data, but did not place any calls within the code before our program was compiled, we can use NightView to insert a *tracepoint* in the

code. A tracepoint is a call to one of the **ntrace(3X)** library routines for recording the time when execution reached the tracepoint.

### To add a tracepoint in a program

- In the NightView Principal Debug Window, click on the line:

data\_process.do\_work;

- Select Set tracepoint... from the Eventpoint menu. This will open the Set a New Tracepoint dialog.

Set a l	Wes Tracepoint
irog.æ.15{	
	Options: Enable Enable, clisable after next hit. Disable
440 <u>2</u>	
Delete	Cancel
	rag.æ15

Figure 1-18. Setting a new tracepoint

- Enter the 4402 for the Event ID. The value 4402 is typically used as the event ID for user trace events in Ada programs. For example, the MAX-Ada utility, **a.trace**, expects user trace events to have an event ID of 4402.
- Enter i in the Value field. This will log the value of the variable i as arg1 in the trace file every time this tracepoint is encountered.
- Press OK.

## NOTE

You may have also entered the following command in the Command field of the NightView Principal Debug Window:

tracepoint 4402 at line\_number value=i

where *line\_number* coincides with the line:

data\_process.do\_work;

See tracepoint for details on the use of this command.

## **Resuming execution**

Now it's time to let the program run and generate some trace data from the tracepoint we just entered.

## To resume execution in NightView

- Press the Resume button in the NightView Principal Debug Window.



Figure 1-19. Resuming execution

## Starting the simulation

Now we need to go back to our NightSim Scheduler window and start the simulation. When you click on the Start button, NightSim carries out the following actions:

- Attaches the timing source to the scheduler if not already attached or if the timing source has been changed
- If a real-time clock is being used as the timing source, sets the clock period in accordance with the value entered in the Clock period field in the Scheduler Configuration Area
- Starts the simulation with the values of the *minor cycle*, *major frame*, and *overrun* counts set to zero

#### To start a simulation in NightSim

- Press the Start button on the NightSim Scheduler window.



Figure 1-20. Starting the simulation

## Inserting a patchpoint

NightView allows the use of *patchpoints* while debugging a process. Patchpoints are locations in the debugged process where a *patch*, usually an expression that alters the behavior of the process, is inserted.

In our example, we will insert a patchpoint in the loop to change the value of the istat variable in order to exit the loop:

```
while istat = 0 loop
   data_process.do_work;
   rt_interface.FBS_wait( istat );
   i := i + 1;
end loop;
```

### To insert a patchpoint in a program

- In the NightView Principal Debug Window, click on the line:

while istat = 0 loop

- Select Set patchpoint... from the Eventpoint menu. This will open the Set a New Patchpoint dialog.

ocation	nmn #13	
Eventpoint Number:	hundline of	Options: U Enable Enable, disable after next hit Uisable
Condition: # Ignare Count:	1	
Vame:	<ul> <li>Insert an exp</li> <li>Brench to a d</li> </ul>	ression at this location Illiferent location
Evaluate:	istat := -1	Inc. Inc.

### Figure 1-21. Setting a new patchpoint

- Enter the expression:

istat := -1

in the Evaluate field.

When this patchpoint is encountered during the execution of the program, the value of the variable istat will be set to -1, breaking out of the loop, thereby terminating the program.

- Press OK.

#### NOTE

You may have also entered the following command in the Command field of the NightView Principal Debug Window:

patchpoint at line\_number eval istat := -1

where *line\_number* coincides with the line:

while istat = 0 loop

See **patchpoint** for details on the use of this command.

## Halting user tracing and kernel tracing

Now that our program has finished, we can exit the KernelTrace utility and stop the Night-Trace user daemon.

#### To halt kernel tracing

- On the PowerMAX OS system where you invoked the KernelTrace utility (see "To activate kernel tracing" on page 1-24), press Cntl-C.

You should see the message:

terminating

#### To halt the NightTrace user daemon

- On the PowerMAX OS system where you invoked the NightTrace user daemon (see "To activate the NightTrace user daemon" on page 1-23), enter the following command:

ntraceud -quit program\_name.trace.data

where *program\_name* is the name of the program generating the trace data. So, for our example, we will issue the following command:

ntraceud -quit prog.trace.data

## **Disabling the patchpoint**

Before we exit NightView, we should disable the patchpoint that we set in "Inserting a patchpoint" on page 1-33. NightView retains knowledge of all eventpoints for a particular program in a current session and will reinitialize them if that program is re-run. If not disabled, the patchpoint in our program will be encountered immediately if our program is re-run under the current session of NightView, causing us to exit the loop and terminate the program.

### To disable a patchpoint in NightView

- Select Summarize/Change... from the Eventpoint menu.
- Select the patchpoint from the list of eventpoints (listed with a P in the Type column).

	547.6	New Patimpoint.
Location	prog.æ13į́	
Eventpoint Number:		Options: U Enable Enable, disable after next h Disable
Condition: If	1	
lgnare Count:	1	
Name:	1	
Funkator	<ul> <li>Insert an exp Branch to a</li> </ul>	pression at this location different location
Crussee.	Contra 1	Cascel Hele
UK	Deleve	Cancel

### Figure 1-22. Disabling a patchpoint

- Press Disable.
- Press Close.

## Exiting the program

NightView suspends the process it is debugging before it exits. We may allow the process to complete its termination by resuming its execution.

## To resume execution in NightView

- Press the Resume button in the NightView Principal Debug Window.

19 * 1 end 20 1	loop;
	Chan   Next
Resume	
Print	Breakpoint
Qualifian	Commond

Figure 1-23. Resuming execution

# Removing the scheduler

## To remove the scheduler

- In the NightSim Scheduler window, press the Remove button.



Figure 1-24. Removing the scheduler

You will be presented with the following dialog:



Figure 1-25. Removing the scheduler

- Press Yes to kill the processes that are currently scheduled on the scheduler.

## Using NightTrace

NightTrace is a graphical tool for analyzing the dynamic behavior of single and multiprocessor applications. NightTrace can log application data events from simultaneous processes executing on multiple CPUs or even multiple systems. NightTrace combines application events with PowerMAX OS events and presents a synchronized view of the entire system. NightTrace allows users to zoom, search, filter, summarize, and analyze events in a wide variety of ways. PowerMAX OS events include individual system calls, context switches, machine exceptions, page faults and interrupts. Application events are defined by the user allowing logging of the data items associated with each event.

We may use NightTrace to analyze the trace data that we gathered during the execution of our program but first we will need to convert the files so that they may be used by Night-Trace.

#### NOTE

If you do not have a system running PowerMAX OS networked to your Linux system, you will need to copy the files contained in the directory **tutorial-sup** from the installation CD to the working directory you created in "Getting Started" on page 1-5.

Proceed to the section titled "Invoking NightTrace" on page 1-41.

## Converting kernel trace event files

#### To convert kernel trace event files

 On the PowerMAX OS system where you invoked the KernelTrace utility (see "To activate kernel tracing" on page 1-24), enter the following command:

ntfilter -v < raw\_kernel\_file > filtered\_kernel\_file

where *raw\_kernel\_file* is the file we specified using the **-o** option to **ktrace** and *filtered\_kernel\_file* is the name of the resultant output file from **ntfilter**.

So, for our example, we will issue the following command:

ntfilter -v < prog.ktrace.data > prog.ntrace.kernel

The converted KernelTrace trace event file will then be saved to the file **prog.ntrace.kernel**. The **-v** option creates a **vectors** files that will be specified to NightTrace along with the converted KernelTrace trace event file. The **vectors** file is generated dynamically because it is system-configuration dependent. Without a **vectors** file, NightTrace will not be able to display the names of the system processes, interrupts, and exceptions that occurred during kernel tracing.

See "Converting KernelTrace Trace Event Files with ntfilter" in the *NightTrace Manual* (0890398) for more detailed information about this process.

## **Creating NightTrace configuration files**

## To create NightTrace configuration files

- On the Linux system, use the MAXAda utility, **a.trace** to create the NightTrace configuration files from the file generated by the NightTrace user daemon. The command has the following syntax:

#### a.trace <program\_name.trace.data</pre>

where *program\_name* is the name of the program that generated the trace data.

So, for our example, we will issue the following command:

#### a.trace prog.trace.data

#### NOTE

You may need to prepend /usr/ada/bin to the a.trace command if you did not add it to your PATH. See "Additions to PATH" on page 1-4 for more information.

This command creates the following two files:

1. program\_name.ntrace.data

This file is a hard link to *program\_name.trace.data*. See for more information about this file.

2. program\_name.ntrace.config

This file contains string tables, format tables, and a NightTrace display page, including descriptions of NightTrace display objects for this application's trace events.

See Creating the NightTrace Configuration File in the *MAXAda Reference Manual* (0890516) for more detailed information about this process.

## Invoking NightTrace

Now that all our files are created and converted, we may invoke NightTrace and analyze the results.

## To invoke NightTrace

- On the Linux system, enter the following command

```
ntrace prog.ntrace.* vectors
```

This will start the NightTrace Analyzer and pass to it:

prog.ntrace.*	the files created by "Converting kernel trace event files" on page 1-39 and "Creating NightTrace configuration files" on page 1-40
vectors	a file created by "Converting kernel trace event files" on page 1-39 which allows NightTrace to display the names of the sys- tem processes, interrupts, and exceptions that occurred during kernel tracing.

See ntrace Arguments for more information about invoking NightTrace.

NightTrace will present the NightTrace window as well as a display page configured using the **prog.ntrace.config** file created in "Creating NightTrace configuration files" on page 1-40. Both windows are shown below:

-	NightTrace	
File	Help	
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 221	NightTrace performance analyzer - Version 4.2 Copyright (C) 2000, Concurrent Computer Corporation 2 trace event log files read. User trace event log file: prog.ntrace.data. 122 trace events plus 4 continuation events. 122 events saved in memory. 0 trace events lost. 81.1551428s time span, from 0.00000000s to 81.1551428s. Kernel trace event log file: prog.ntrace.kernel. 61320 trace events plus 53090 continuation events. 61320 events saved in memory. 0 trace events lost. 23.1248402s time span, from 47.6431822s to 70.7680224s. Time Base Register was used to time stamp events. 61442 total events read from disk plus 53094 continuation events. 61442 total events saved in memory.	

### Figure 1-26. NightTrace Main window

For more information on the NightTrace window, see ntrace Global Window in the *NightTrace Manual* (0890398).

File Edit Greate Canfigure Expressions Tools - Edit = View	Scotlinger; progettinger, and is s Help	
Task News, "Lepid, "task_id   Eds Events (pury 	de = ell tasks : ref = individual tasks)	Ada Event Description
Time Start 0.000000 Event Start 0 Zooin Factor 2.0 Apply Reset Conter	Time Length 4,057/5724 Event Count 7:3 Increment 5:00 Mark Zoon Region	Time End 4,8575524 Event End 28 Current Time 2,638766 Zoom in Zoom Out Refresh

### Figure 1-27. NightTrace display page

For more information on display pages, see The Display Page in the *NightTrace Manual* (0890398).

## Creating a default kernel page

In order to view our kernel trace events, we need to create a default kernel page.

### To create a default kernel page

- In the NightTrace window, select Default Kernel Page from the File menu.

This will create a Default Kernel Page as shown below:

	RightTrace; strattering	
le Edit Greate Canfigure Expressions	Tools Help	
> Edit • View		
(B) a		
CPU 1		
Interrupt Econotice Special	2,8 	2.0
******		
) <sup>(1)</sup>		
Time Start (0,000000) Event Start (0 pom Fadar (2,9	Time Length (4,057572) Event Count [35 Increment [35,003	Time End 4,857552s Event End 38 Current Time 2,83876s

### Figure 1-28. Default Kernel Page

For more information on the Default Kernel Page, see Kernel Display Pages in the *Night-Trace Manual* (0890398).

## Searching for a kernel trace event

Now that we have loaded our data into NightTrace and created the appropriate display pages, we can search for the system call that corresponds to the FBS\_wait call made in our program (see "Using NEdit" on page 1-5).

#### To search for a kernel trace event

- Select Search... from the Tools menu of the kernel display page (see "Creating a default kernel page" on page 1-42).

You will be presented with the following dialog:

-	Search 🗉 🔲	
Search		
Search Direction:	Search Constraints:	
<ul> <li>Forward</li> </ul>	Global Search	
🔶 Backward	$\diamond$ Interval Search	
Inte	erval Manipulation:	
<ul> <li>Scrol</li> </ul>	I Current Time to Event	
🔷 Zoon	n to Include Event	
◇ Do N	ot Move Current Time	
Event List TR	SYSCALL_RESUME	
No Event List NOM	IE III	
If Expression arg2 == get_item(syscall, "fbswait")		
CPU List ALL		
PID List ALL		
TID List ALL		
Apply Reset Prev Next Search Close		

Figure 1-29. Searching for a kernel trace event

- Enter TR\_SYSCALL\_RESUME in the Event List field. This trace event is logged whenever a system call (syscall) is resumed (i.e., the process that caused the syscall to occur, which was switched out before the syscall could be completed, is switched back in).
- Enter arg2 == get\_item(syscall, "fbswait") in the lf Expression field. The fbswait system call corresponds to the FBS\_wait call we made in our Ada program.
- Press Apply.
- Press Search.

NightTrace will set the current time to that of the first logged kernel trace event that matches the specified search criteria, positioning the grid on the kernel display page accordingly. This is shown in the figure below. Note the Current Time. In our example, it is set to 48.5713587 seconds.

### NOTE

Since we specified the **-bufferwrap** option to **ktrace** (see "To activate kernel tracing" on page 1-24), it is likely that the earlier trace events may have been overwritten by buffer wraparound during the execution of the program. Hence, we may not actually see the *first* actual kernel trace event that corresponds to our search criteria. However, this is sufficient for our example.

Re Edit Create Canfigure Expr	essions Tools Help
<ul> <li>Edit</li> <li>1 Search criteria est at</li> <li>View</li> </ul>	Zülth east.
CPU s pid prog Filmwait	
4	
Time Start 44,5434401a Event Start 34 Zoom Factor 2,9	Time Length 4,057852s Time End 30,0000378s Event Count 4984 Event End 5087 Indrement 30,000 Current Time 48,0513987s
Apply Feset	Center Mark Zoom Region Zoom In Zoom Out Refresh

Figure 1-30. First kernel trace event

In addition to setting the current time and repositioning the grid on the kernel display page when the search for the kernel trace event was performed, NightTrace will automatically set the current time and reposition the display page that contains the user trace events as well. This is shown in the figure below.

Re Edit Create Configure Expression	is Tools Help	
- Edit • View		
Terk News, "Legid, "task_id Hds Ev	nts (purple = all tasks : red = individual tasks)	Rde Event Description
Durrent Taski 'L 'L 'L	······································	oor.Jracoi oub.id=0x3e, data1=0x3a, data2=0x
Senvironment> 1 1		ser_trate: sub_id=Nv3e, data1=Nv3a, data2=Nv
c		
Time Start 46,5424001s Event Start 34 Zoom Factor 2.0	Time Longth (4,057572) Event Court (4994 Increment (25,062	Time End 50,5002775 Event End 5067 Current Time (45,5713507)
Acoby   Reset	Center   Mark   Zoon Region	Zoom In Zoom Out Refresh

Figure 1-31. NightTrace display page repositioned accordingly

## Searching for a user trace event

Now that we have found the first logged kernel trace event, we can search for the user trace events that we logged using NightView (see "Adding a tracepoint in the program" on page 1-30).

### To search for a user trace event

### NOTE

You may use the same search dialog that you used in the previous step, "Searching for a kernel trace event" on page 1-43.

- Select Search... from the Tools menu of the display page created when NightTrace was invoked (see "Invoking NightTrace" on page 1-41).

You will be presented with the following dialog:

-	Gearch 🗖	
Search		
Search Direction:	Search Constraints:	
<ul> <li>Forward</li> </ul>	Global Search	
🔷 Backward	$\diamond$ Interval Search	
Interva	al Manipulation:	
<ul> <li>♦ Scroll Current Time to Event</li> <li>♦ Zoom to Include Event</li> <li>♦ Do Not Move Current Time</li> </ul>		
Event List 4402		
No Event List NONE		
If Expression TRUE		
CPU List ALL		
PID List ALL		
TID List ALL		
Apply Reset Prev Next Search Close		

### Figure 1-32. Searching for a user trace event

- Enter 4402 in the Event List field. This corresponds to the Event ID for the tracepoint we specified in NightView (see "Adding a tracepoint in the program" on page 1-30).
- Ensure that the value of the If Expression field is TRUE.

- Press Apply.
- Press Search.

NightTrace will set the current time to the first user trace event after the current time that matches the specified search criteria, positioning the grid on the kernel display page accordingly. This is shown in the figure below. Note the Current Time now. In our example, it is set to 48.5714136 seconds, 0.0000549 seconds after the fbswait system call we found in "Searching for a kernel trace event" on page 1-43.

You can alternately search between the kernel display page (see "To search for a kernel trace event" on page 1-43) and the display page which contains the user trace events (see "To search for a user trace event" on page 1-46) to see that an fbswait system call always precedes the user trace event that we logged, which is what we would expect.

### NOTE

If you used the same search dialog as you used for searching for a kernel trace event, you may use the **Prev** button on the search dialog for the previous search criteria. You can alternate between searching for user trace events and kernel trace events using this functionality.

File Edit Create Canfigure Expr	RightTrave; progetrave.cov/jg essions Tools Help
- Edit 1 Search criteria set at * View	2534th want.
Task News, 'Lapid, 'task_id	Rde Events (purple = ell tasks : red = individual tasks) Rde Event Description
Convent Taski 1 1 1	147.9
Senvironment> 1 1	weer_Srace1 sub_1640x3b, data140x3b, data240x
4	
Time Start 46,5424003x Event Start 54 Zoom Factor 2.0	Time Length (4.057572)         Time End (50.000373)           Event Count (484         Event End (507           Increment (5.067         Current Time (4.5714)%
Apply Reset	Center Mark Zoom Region Zoom In Zoom Out Refresh

Figure 1-33. NightTrace display page

## Zooming in

### To zoom in:

- You may use the Zoom In button on the NightTrace Analyzer to see more details.

For our example, we zoomed in on our kernel display page 13 times to see the following level of detail.

Re Edit Create Canifigure Expressions Tools Help         Edit       1       Search criteria net at 2502kt event,         View       2       Search criteria net at 2502kt event,         View       1       Invalid         Picinfir       Invalid       Invalid         Internet       Event Invalid       Invalid         Picinfir       Invalid       Invalid         Internet       Event Invalid       Invalid         Invalid       Event Invalid       Invalid         Invalid       Event Invalid       Invalid         Invalid       Event Invalid       Event Invalid         I		HogenTracet stractISF8d
Edit       1       Sauch grutaria set at 2020k event,         View       2       Sauch grutaria set at 2010k event,         View       1       Sauch grutaria set at 2010k event,         (Pi a)       Immail       Immail         (Pi 1)       Immail       Immail         <	Re Edit Create Configure Expressions	s Tools Help
(R) 8     Fierdelook       pid prog     Fierdelook       (P) 1     Pierdelook       pid side     Pierdelook       (P) 1     Pierdelook       pid side     Pierdelook       (P) 1     Pierdelook       (P) 2     Pierdelook <tr< th=""><th><ul> <li>Edit         <ol> <li>Search criteria set at 2534th</li> <li>Search criteria set at 2534th</li> </ol> </li> </ul></th><th>exert.</th></tr<>	<ul> <li>Edit         <ol> <li>Search criteria set at 2534th</li> <li>Search criteria set at 2534th</li> </ol> </li> </ul>	exert.
Time Start 40.5711894:         Time Length 0.004454:         Time End 40.571897:           Event Start 369         Event Count 9         Event End 2507           Zoon Factor 2.4         Increment 3.047         Ourrent Time 40.5714336           Apply         Reset         Center         Mark         Zoon Region         Zoon in         Zoon Out         Refresh	CPU s pid prog Filmealt	43,57126 43,57236 45,57546 49,57156 48,57156
Time Start         Fine Length         OxiVerSHs         Time End         Event Start           Event Start         259         Event Count 9         Event End         Event End         Event End           Zoan Fadar         2.4         Increment         3.04         Current Time 49.97(4)%           Apply         Reset         Center         Mark         Zoam Region         Zoam In         Zoam Out         Refresh	4	7
Apply Reset Center Mark Zoom Region Zoom In Zoom Out Refresh	Time Start 44,5711834 Event Start 25/9 Zoom Factor 2,9	Time Length 0.004844         Time End 40.0010804           Event Count (9         Event End (307           Increment (3004)         Current Time 40.0014344
and and an and a second	Apply Reset Center	r Mark Zoom Region Zoom In Zoom Out Refrest

Figure 1-34. Zoomed in kernel display page

In the above figure, the first bar (red) indicates the real-time clock interrupt for this cycle. The second bar (blue) shows the target process **prog** exiting the FBS\_wait call in the Ada code. The current time line is positioned at the user trace event that we previously searched for.

Looking at the other display page (which shows our user trace events), we can see the user\_trace event inserted through NightView (see "Adding a tracepoint in the program" on page 1-30). Note that both displays are synchronized in time (the current time line represents the same instant in time on both display pages). You may middle-click on the line representing the user trace event to see more detailed information.

The third and final bar (blue) on the kernel display page represents the next encounter of the FBS\_wait call in the loop.

### NOTE

Due to a problem with the **-bufferwrap** option to the **ktrace** command, user and kernel data may not appear synchronized. This problem has been fixed in the **ktrace** and **ntfilter** commands in PowerMAX OS 4.3 Patch Set 6 (**trace-004** and **base-006**). See "To activate kernel tracing" on page 1-24 for more information.

# Conclusion

This concludes our tutorial for the PowerWorks Linux Development Environment. We hope that we have given you a sufficient overview of the various tools and the interactions between them.

PowerWorks Linux Development Environment Tutorial

Product Name: 0.5" from top of spine, Helvetica, 36 pt, Bold

Volume Number (if any): Helvetica, 24 pt, Bold

Volume Name (if any): Helvetica, 18 pt, Bold

Manual Title(s): Helvetica, 10 pt, Bold, centered vertically within space above bar, double space between each title

Bar: 1" x 1/8" beginning 1/4" in from either side

Part Number: Helvetica, 6 pt, centered, 1/8" up

Development Environm Powerv S Tutorial 0898100