LynuxWorks Development Tools CodeWarrior IDE Edition User's Guide

Release 1

DOC-0393-01

Contents

Preface

This document contains information on configuring and using LynuxWorks Development Tools, CodeWarrior Integrated Development Environment Edition.

For More Information

For additional information pertaining to LynuxWorks Development Tools, CodeWarrior IDE Edition, refer to the following printed and online documentation.

- *LynuxWorks CodeWarrior IDE Edition Release Notes*

  This document contains installation instructions, late-breaking information, and known limitations for this release.

- *LynxOS Installation Guide*

  This manual supports the initial installation and configuration of LynxOS and the X Windows System.

- *LynxOS User's Guide*

  This document contains information about basic system administration and kernel level specifics of LynxOS. It contains a “Quick Starting” chapter and covers a range of topics, including tuning system performance and creating kernel images for embedded applications.

- *LynxOS Release Notes*

  This document contains late-breaking information for LynxOS.

- *BlueCat Linux User's Guide*

  Contains information on installation and use of BlueCat Linux

- Online information

CodeWarrior IDE Edition documentation provided by Metrowerks is included with the CodeWarrior product in HTML format. The online help system can be accessed by clicking the **Help** menu button.

For more information on using CodeWarrior IDE for developing native applications for Linux or Solaris, see the MetroWerks online documentation or website (www.metrowerks.com).

Typographical Conventions

The typefaces used in this manual, summarized below, emphasize important concepts. All references to filenames and commands are case sensitive and should be typed accurately.

Font and Description	Examples
Italicized; used for book titles, text representing a variable, function names and new terms.	Refer to the *LynxOS User's Guide*. mv *`file1 file2`* *`getenv()`*
Bold; Keyboard options, button names, menus, and command lines and options entered on the computer.	**Enter, Ctrl-C** Click **OK** **Help** –> **About LynuxWorks** `#` **`ls -l`**

Special Notes

The following notations highlight any key points and cautionary notes that may appear in this manual.

NOTE: These callouts note important or useful points in the text.

CAUTION! Used for situations that present minor hazards that may interfere with or threaten equipment/performance.

Technical Support

LynuxWorks Technical Support is available Monday through Friday (holidays excluded) between 8:00 AM and 5:00 PM Pacific Time (U.S. Headquarters) or between 9:00 AM and 6:00 PM Central European Time (Europe).

The LynuxWorks World Wide Web home page provides additional information about our products, Frequently Asked Questions (FAQs), and LynuxWorks news groups.

LynuxWorks U.S. Headquarters

Internet: **`support@lnxw.com`**
Phone: (408) 979-3940
Fax: (408) 979-3945

LynuxWorks Europe

Internet: **`tech_europe@lnxw.com`**
Phone: (+33) 1 30 85 06 00
Fax: (+33) 1 30 85 06 06

World Wide Web

`http://www.lynuxworks.com`

CHAPTER 1 Overview

LynuxWorks Development Tools, CodeWarrior Integrated Development Environment (IDE) Edition is a fully-featured development environment that supports all LynuxWorks cross-development platforms. This software combines the quality of LynuxWorks cross-development tools with a single integrated development environment provided by Metrowerks' CodeWarrior IDE.

The LynuxWorks Development Tools, CodeWarrior IDE Edition provides an intuitive user interface that facilitates editing and debugging source code. The creation of complicated makefiles can be automated with drag-and-drop functionality. If a command line interface is preferred, developers can use the CodeWarrior IDE to manage and create new projects from UNIX makefiles.

The LynuxWorks CodeWarrior IDE Edition product includes LynuxWorks "stationery" projects that allow users to create applications based on sample template code. This template code allows users to start projects with fully functional code that can be compiled and executed.

CHAPTER 2

Developing Applications

This chapter describes the process of developing and debugging applications with LynuxWorks Development Tools, CodeWarrior IDE Edition.

Development Process Overview

The following figure details the key steps of the LynuxWorks Development Tools, CodeWarrior IDE Edition development process:

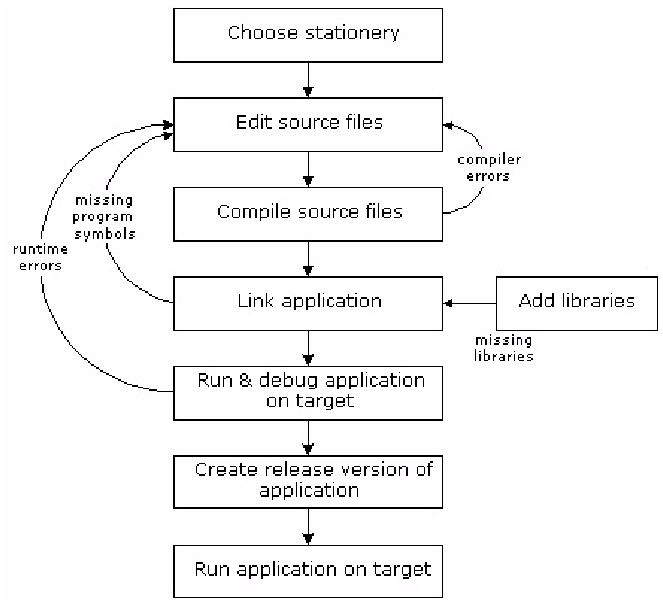


Figure 2-1: The Development Process Chart

Stationery Files

LynuxWorks Development Tools, CodeWarrior IDE Edition includes a number of sample projects, also known as project stationery files. These stationery files provide examples of pre-configured settings and application source code.

Stationery files allow users to start the development process with a project similar to their own.

Each stationery file includes two targets for each supported target operating system and architecture combination. The first target option (_D) builds a debug version of the application. The second target (_R) creates a release version, with the debug information removed.

The following table describes the available stationery files included with LynuxWorks Development Tools, CodeWarrior IDE Edition:

Table 2-1: Available Stationery Files

Stationery file	Description
`C++_Hello_World_Application`	Simple C++ application
`C_Hello_Application`	Simple C application
`C_CURSES_Application`	C application that uses the `libncurses` library
`C_Condition_Variable_Mutex_Application`	Multi-threaded C application that uses condition variables and `mutexes`
`C_Fork_Exec_Application`	C application that shows use of the `fork()` and `exec()` functions
`C_Interval_Timers_Application`	C application that uses POSIX1b facilities for creating, reading, and deleting an interval timer
`C_POSIX_Message_Queues_Application`	C application that uses message queues. This stationery is supported for LynxOS targets only.
`C_POSIX_Multithreaded_Application`	Multi-threaded C application
`C_POSIX_Semaphores_Application`	C application that uses semaphores to synchronize two processes

Table 2-1: Available Stationery Files (Continued)

Stationery file	Description
`C_POSIX_Shared_Memory_Application`	C application that uses the `shm_open` function the POSIX1b `mmap` facility. This stationery is supported for LynxOS targets only.
`C_POSIX_Signal_Application`	C application that uses POSIX signal handling functions
`C_Priority_Application`	C application that modifies priority of a child process
`C_Realtime_Events_Application`	C application that uses POSIX1b real-time event facilities

Creating a Custom Project from a Stationery File

To create a custom project from a LynuxWorks stationery file:

1. Select **File -> New** from the **Project** toolbar.
2. Click the **Project** tab in the **New** window to display the available stationery sets.
3. Click **LynuxWorks Stationery** in the **Project** tab.
4. Enter a project name in the **Project Name** field. Be sure to add the **`.mcp`** suffix explicitly.

   To change the location of the project, type the full path to the project file in the **Location** field, or use the **Set** button to select the directory. Clicking **Set** opens a window to navigate through the system's directories.
5. Click the **OK** button.

The following figure is an example of the **New** window:

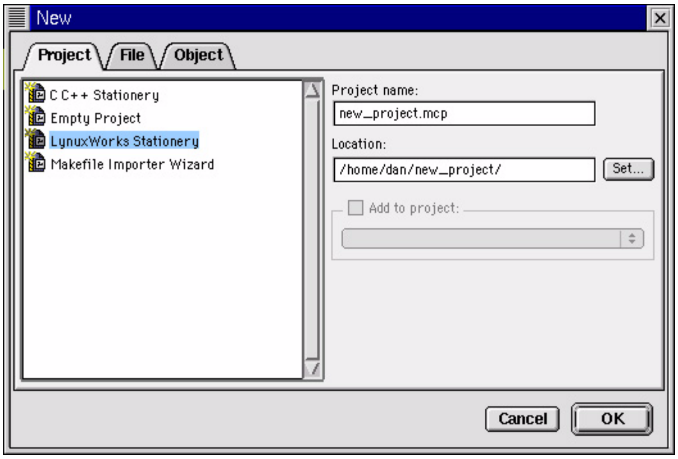
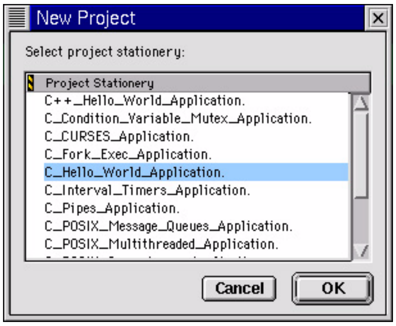


Figure 2-2: The New Window

As a result, the **New Project** window appears, listing sample projects. The following figure details the **New Project** window:



C++_Hello_World_Application.
C_Condition_Variable_Mutex_Application.
C_CURSES_Application.
C_Fork_Exec_Application.
C_Hello_World_Application.
C_Interval_Timers_Application.
C_Pipes_Application.
C_POSIX_Message_Queues_Application.
C_POSIX_Multithreaded_Application.

Figure 2-3: The New Project Window

6. Select a sample project (stationery file).
7. Press the **OK** button to complete the new project setup.

The new project is configured with the following information from the stationery file:

- Source files and libraries
- Dependencies
- Preferences and target settings

LynuxWorks Development Tools, CodeWarrior IDE Edition creates a new project based on the selected stationery file and automatically opens it.

NOTE: If cross development tools are not installed on the development host, warning messages appear that the build target cannot be found. If the specified build targets are not required for development, these messages can be ignored. If these cross-development tools are required, ensure that they are installed properly on the host system.

Editing Source Files

Source files can be edited in LynuxWorks Development Tools, CodeWarrior IDE Edition with either the default CodeWarrior IDE screen editor or the vi text editor.

Using the vi Editor

For information on using the vi editor, see the `vi(1)` man page. To set vi as the default editor:

1. Select the **Edit -> Preferences**.
2. Select the **IDE Extras** panel.
3. Click the **Use External Editor** checkbox in the **IDE Extras** panel.

Using the CodeWarrior IDE Editor

The CodeWarrior IDE editor can automatically verify the balance of parentheses, brackets, and braces. Pop-up context menus provide quick navigation to the beginning of functions, or header files. C/C++ keywords are color-coded for ease of recognition and navigation. Custom keyword sets can also be defined.

The CodeWarrior IDE editor uses a graphic difference engine that can display the differences between two source code files. The contents of the two files are displayed in a split-screen, with differences marked in color.

To open a project source file for editing:

1. Click the **File** tab in the **Project** window to view the source files.
2. Open the **Source** folder and double click on the appropriate file to open it.

The following figure illustrates the selection of a source file:

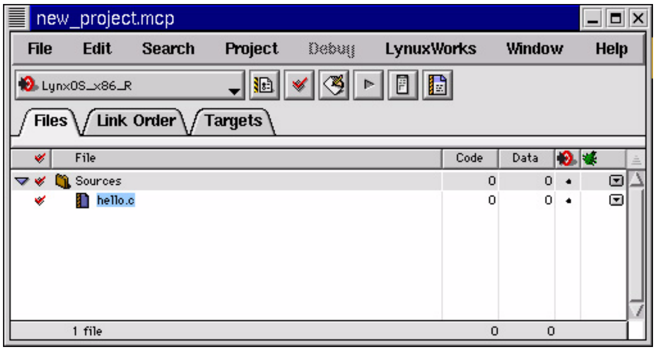
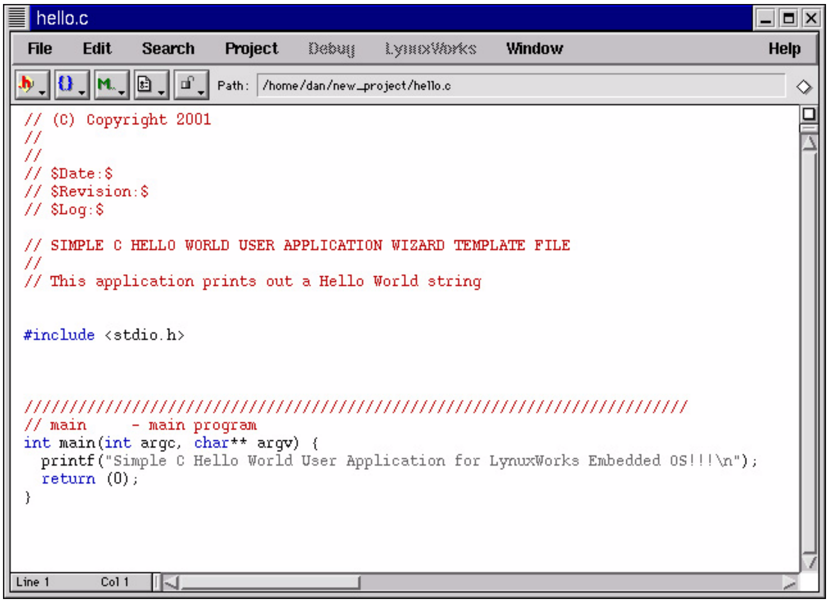


Figure 2-4: Selecting a Source File

The file is opened in the **Editor** window. The following figure shows the default CodeWarrior IDE **Editor** window:



```
// (C) Copyright 2001
//
//
// $Date:$
// $Revision:$
// $Log:$

// SIMPLE C HELLO WORLD USER APPLICATION WIZARD TEMPLATE FILE
//
// This application prints out a Hello World string


#include <stdio.h>



///////////////////////////////////////////////////////////////////////////
// main      - main program
int main(int argc, char** argv) {
  printf("Simple C Hello World User Application for LynuxWorks Embedded OS!!!\n");
  return (0);
}
```

Figure 2-5: A Sample Editor Window

Building a Project

The LynuxWorks Development Tools, CodeWarrior IDE Edition provides access to LynuxWorks cross-development tools. It supports the creation of executables for different target architectures from a single project. The appropriate toolset must be selected for each target before the project can be built.

Use the following procedure to select an appropriate toolset for a target:

1. Open the **Target Settings** panel from the **Edit** menu, where **Target** is the name of the currently selected target. For example, select **LynxOS_x86_R Settings** from the **Edit** menu.
2. Click the **Linker** button in the **Target Settings** panel to change the Linker.
3. Select a linker.

   The following linkers are supported on the host:

- **GCC Linker**
  Native host linker
- **GCC LynxOS X86 Linker**
  LynxOS x86 cross development tools
- **GCC LynxOS PPC Linker**
  LynxOS PowerPC cross development tools
- **GCC LynxOS MIPS Linker**
  LynxOS MIPS cross development tools
- **GCC BlueCat ARM Linker**
  BlueCat Linux ARM cross development tools
- **GCC BlueCat PPC Linker**
  BlueCat Linux PowerPC cross development tools
- **GCC BlueCat SH Linker**
  BlueCat Linux SH cross development tools
- **GCC BlueCat X86 Linker**
  BlueCat Linux x86 cross development tools

The following figure shows the **Target Settings** panel:

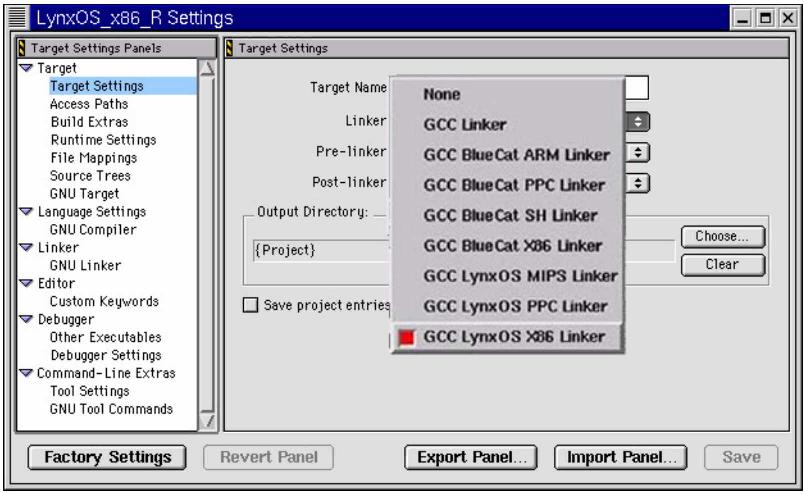


Figure 2-6: The Target Settings Panel

Selecting the linker in the **Target Settings** panel also selects the cross-development tools, such as the compiler and assembler, as well as default options.

Once the Linker is selected, follow these steps to build a project:

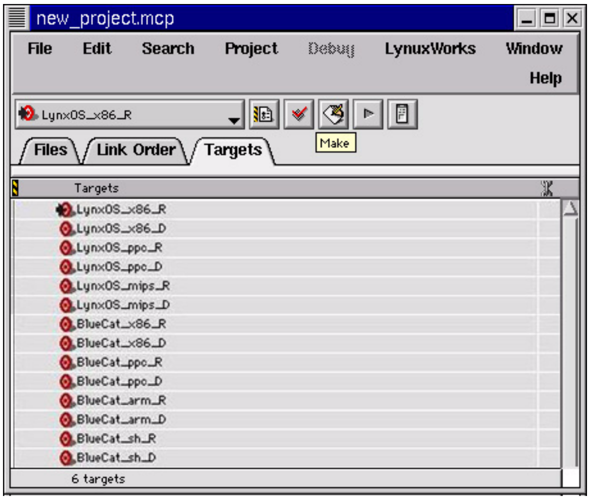
1. Select the **Targets** tab in the **Project** window.

   The list of available targets is displayed.

2. Choose the debug target (debug targets are appended with `_D`).

3. Click the **Make** icon on the toolbar.

The following figure shows the **Targets** view:

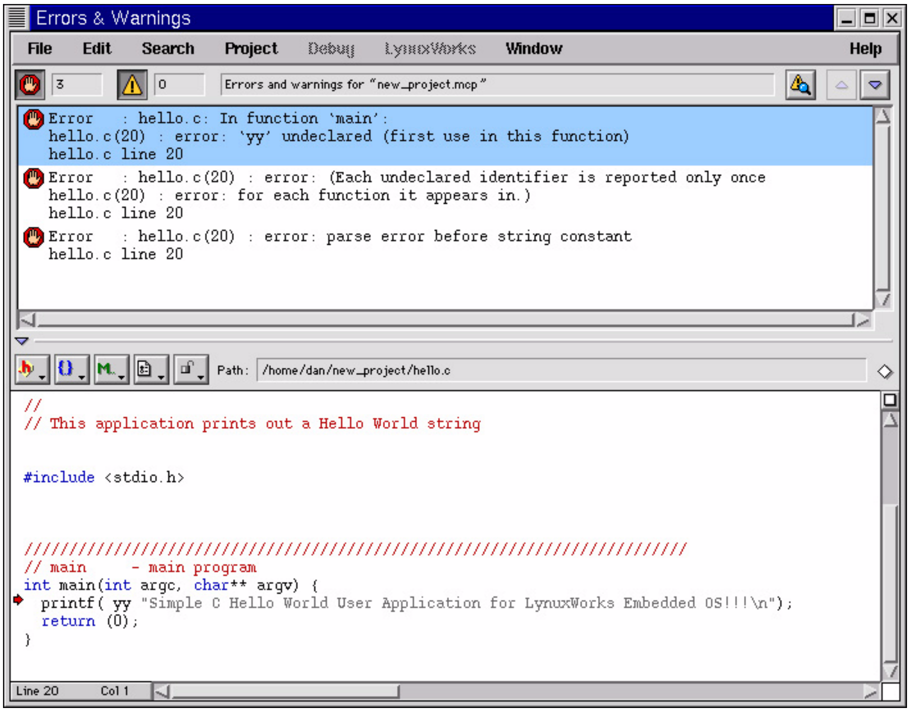


LynxOS_x86_R
LynxOS_x86_D
LynxOS_ppc_R
LynxOS_ppc_D
LynxOS_mips_R
LynxOS_mips_D
BlueCat_x86_R
BlueCat_x86_D
BlueCat_ppc_R
BlueCat_ppc_D
BlueCat_arm_R
BlueCat_arm_D
BlueCat_sh_R
BlueCat_sh_D
6 targets

Figure 2-7: Targets View

Compile time errors are displayed in the **Message** window. The **Message** window can be used to navigate to errors and warnings. To move to the source code line where the compiler detected an error, double-click the error message.

The following figure shows the **Error & Warnings** window:



```
Error   : hello.c: In function 'main':
hello.c(20) : error: 'yy' undeclared (first use in this function)
hello.c line 20
Error   : hello.c(20) : error: (Each undeclared identifier is reported only once
hello.c(20) : error: for each function it appears in.)
hello.c line 20
Error   : hello.c(20) : error: parse error before string constant
hello.c line 20
```

```
//
// This application prints out a Hello World string


#include <stdio.h>



//////////////////////////////////////////////////////////////////////////
// main     - main program
int main(int argc, char** argv) {
  printf( yy "Simple C Hello World User Application for LynuxWorks Embedded OS!!!\n");
  return (0);
}
```

Figure 2-8: The Error & Warnings Window

Debugging Applications

Before debugging can begin, an application must be compiled, linked, and downloaded to the target. Users can debug applications with the supported debugging tools GDB or Total/db. The host uses TCP/IP to communicate with the target. The target system network settings must be properly configured to be able to communicate with the host system. Other target system settings, including GDB server connection type and application target directory, are configured in the **Target System** panel of the **LynuxWorks** menu on the host system.

Refer to Chapter 3, “LynuxWorks Menu Items” on page 19 for descriptions of the **LynuxWorks** menu buttons and the **Target System** panel fields.

The following provides an example of the general procedures used when debugging applications on the target:

1. Open the **Target System** panel from the **LynuxWorks** menu and enter the appropriate target system configurations.
2. Click the **TotalDB** button from the **LynuxWorks** menu to rebuild the current project executable (if necessary), download the current project executable to the target system, and start a Total/db session.
3. Debug the project, setting up necessary breakpoints.

Building a Release Version

When debugging is complete, the release version of the application can be created.

To build a release version of the application:

1. Select the **Targets** tab in the **Project** window.
2. Choose the release target (release targets are appended with `_R`).
3. Click the **Make** icon on the toolbar.

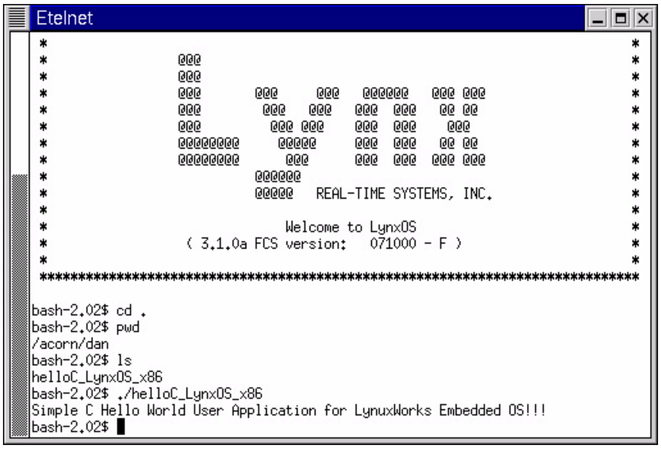
The application is recompiled without the debug information.

Running the Application on the Target

To run an application on the target:

1. Download the executable on the target system using one of the **LynuxWorks** menu buttons (**Copy Executable**, **Ftp** or **Build and Copy Executable**).
2. Run the application on the target from a telnet session or using the **Target Command Execution** menu button.

The following figure shows the result of running an application on the target.



```
*                                                                            *
*                 @@@                                                        *
*                 @@@                                                        *
*                 @@@       @@@     @@@   @@@@@@   @@@ @@@                   *
*                 @@@        @@@   @@@   @@@  @@@   @@ @@                    *
*                 @@@         @@@ @@@    @@@  @@@    @@@                     *
*                 @@@@@@@@     @@@@@     @@@  @@@   @@ @@                    *
*                 @@@@@@@@      @@@      @@@  @@@  @@@ @@@                   *
*                           @@@@@@                                           *
*                           @@@@@   REAL-TIME SYSTEMS, INC.                  *
*                                                                            *
*                               Welcome to LynxOS                            *
*                  ( 3.1.0a FCS version:   071000 - F )                      *
*                                                                            *
******************************************************************************

bash-2.02$ cd .
bash-2.02$ pwd
/acorn/dan
bash-2.02$ ls
helloC_LynxOS_x86
bash-2.02$ ./helloC_LynxOS_x86
Simple C Hello World User Application for LynuxWorks Embedded OS!!!
bash-2.02$ █
```

Figure 2-9: Running an Application

Creating a Hello World Application -- Example

The following sections provide a step-by-step example of creating and debugging a `helloworld.c` file.

Open the helloworld.c Stationery File

Create a new CodeWarrior project using the LynuxWorks stationery file.

1. From the CodeWarrior Project Toolbar, select **File -> New**.
2. Click the **Project** tab in the **New** window to display the available stationery sets.
3. Click the **LynuxWorks Stationery** in the **Project** tab pane
4. Type a name in the project name field. For example, **`helloworld.mcp`** and click **Ok**.
5. In the **New Project** pop-up window, select **C_Hello_World_Application**. and Click **Ok**.

   A new CodeWarrior project is created with the `helloworld.c` source file.

Target System Settings

Before compiling `helloworld.c`, users must first set up the CodeWarrior cross development environment. Follow these instructions to set the appropriate environment. This example assumes the target system is an x86 LynxOS platform, and is connected via ethernet.

Choosing the Cross Development Linker

The Target System Settings (in this example called LynxOS_x86_D Settings), is located in the **Edit** menu of the **helloworld.mcp** project window. Select the appropriate cross development linker with the following steps:

1. Open the **Target Settings** window by clicking on **Edit-> LynxOS_x86_D Settings**.
2. In the **Linker** pane of the **Target Settings** window, select the **GCC LynxOS X86 Linker**.
3. Click **Save** and close the **Target Settings** window.
4. From the **helloworld.mcp** project window, select the **Targets** tab and the **LynxOS_x86_D target**. An arrow should appear next to the **bullseye** icon after clicking. This specifies that the project is compiled for LynxOS x86 systems, with debugging information.

Setting Target System Properties in CodeWarrior

Before using any of the LynuxWorks menu items, users must set the **Target System** properties in the **LynuxWorks** menu.

1. From the **LynuxWorks** menu, select **Target System**.
2. At a minimum, set the target system LAN address (or host name if DNS is configured), user account, password, and GDB settings (for debugging).
3. Click **Save** when finished.

Enabling Remote access on the Target System

To enable remote access on the target, the target system user account must include an `.rhosts` file. The `.rhosts` file specifies remote host names and user names that can access the system. For the user account `guest` on the host system `dev_host`, edit the `.rhost` file as:

```
trustedhost   trusteduser
dev_host      guest
```

Compiling helloworld.c

After the target system settings are set, use the **Build and Copy** command to compile `helloworld.c` and copy the executable to the target.

1. From the **LynuxWorks** menu, select **Build and Copy Executable**. The `helloworld.c` file is compiled, and copied to the target system. An `rcp` window briefly displays.
2. To run the executable, open a telnet window to the target by selecting **LynuxWorks -> Telnet**.
3. In the telnet window, run the hello world program (`helloC_LynxOS_x86`) with the following command:

```
bash-2.02$ ./helloC_LynxOS_x86
```

Debugging helloworld.c

In this example, Total/db is used to debug the hello world application on the target.

1. Start Total/db by clicking **LynuxWorks -> TotalDB**.

   Three windows appear on the host:

   gdb_server (xterm), TDB (xterm), and Total/db.
2. Open the source file by selecting the `helloworld.c` file from the bottom left pull-down menu.
3. Set breakpoints by clicking on any tick marks (`-`) in the column left of the source. A red box appears next to the line at which the break is to occur.
4. Step through the program by clicking the **Run** icon.
5. When the program reaches the first breakpoint, step through the rest of the application by clicking the **Step** icon.
6. Close the Total/db window when finished.

CHAPTER 3

LynuxWorks Menu Items

This chapter describes the **LynuxWorks** menu items.

Overview

The **LynuxWorks** menu provides a quick and convenient way to access the target system. This menu contains buttons to start debug utilities, run applications on the target system, execute commands on the target, and view target system activity. The **LynuxWorks** menu is enabled when a project is opened.

The following figure illustrates the **LynuxWorks** menu items:

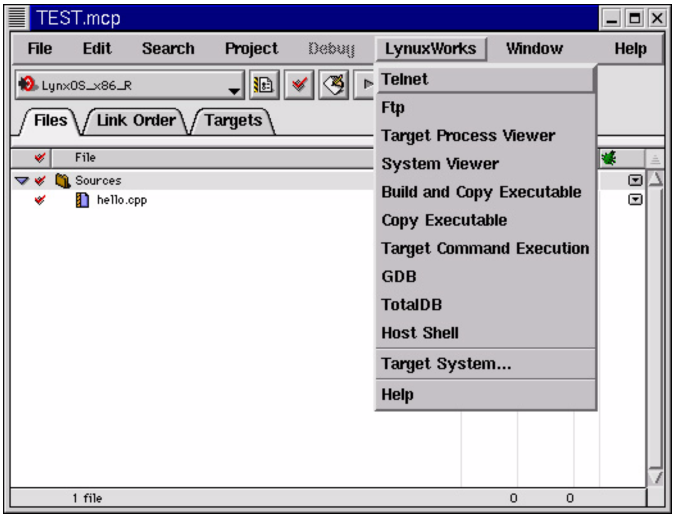


Figure 3-1: LynuxWorks Menu Items

Accessing the Target System

Remote access must be enabled on the target system before users can use the **LynuxWorks** menu items. The host must be able to use standard facilities, such as `rsh` or `rlogin` to access the target system. The host user IDs and passwords must be compatible with the target system. There are several ways to configure a target to accept commands from a remote host. Typically, an `.rhosts` file on the target system is used to list trusted users and system names that can access the target.

To enable remote access, the `.rhosts` file in the user's home directory on the target system must be modified to include the host system name and user name.

The following is an example of an `.rhosts` file:

```
trustedhost  trusteduser
lynuxworks5  test
devhost      user1
```

LynuxWorks Menu Items

This section describes the **LynuxWorks** menu items. Clicking a button in the **LynuxWorks** menu activates a specific function of the LynuxWorks Development Tools, CodeWarrior IDE Edition product. In some cases, additional settings in the **Target System** panel are required. Refer to "Target System Panel" on page 32 for details on setting up the target system.

The following sections describe the buttons available in the **LynuxWorks** menu.

Telnet

Provides telnet access to the target system.

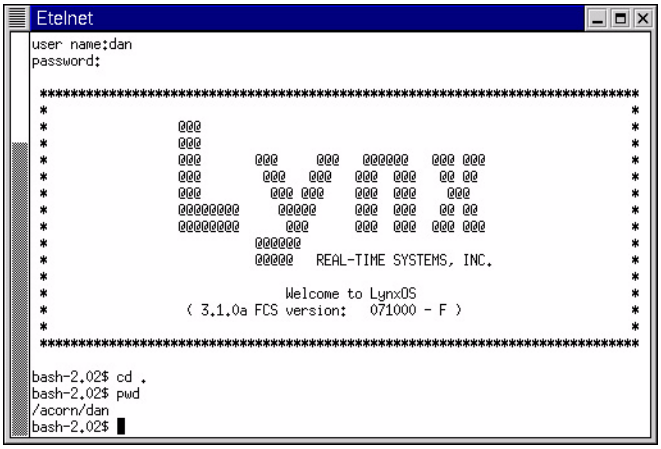
Click on the **Telnet** button to establish a telnet connection with the target and change to the Telnet Target Directory.

To configure **Telnet**, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Password
- Telnet Port
- Telnet Target Directory

NOTE: If the Telnet Target Directory is empty in the **Target System** panel, the Target System Directory field is used.

The following figure illustrates a sample telnet session activated by clicking the **Telnet** button:



```
user name:dan
password:

 *********************************************************************************
 *                                                                               *
 *                 @@@                                                           *
 *                 @@@                                                           *
 *                 @@@        @@@     @@@   @@@@@@    @@@ @@@                    *
 *                 @@@         @@@   @@@   @@@  @@@    @@ @@                     *
 *                 @@@          @@@ @@@    @@@  @@@     @@@                      *
 *                 @@@@@@@@      @@@@@     @@@  @@@    @@ @@                     *
 *                 @@@@@@@@       @@@      @@@  @@@   @@@ @@@                    *
 *                            @@@@@@                                             *
 *                            @@@@@   REAL-TIME SYSTEMS, INC.                    *
 *                                                                               *
 *                                Welcome to LynxOS                              *
 *                   ( 3.1.0a FCS version:   071000 - F )                        *
 *                                                                               *
 *********************************************************************************

bash-2.02$ cd .
bash-2.02$ pwd
/acorn/dan
bash-2.02$ █
```

Figure 3-2: Telnet Session

Ftp

Provides FTP access to the target system.

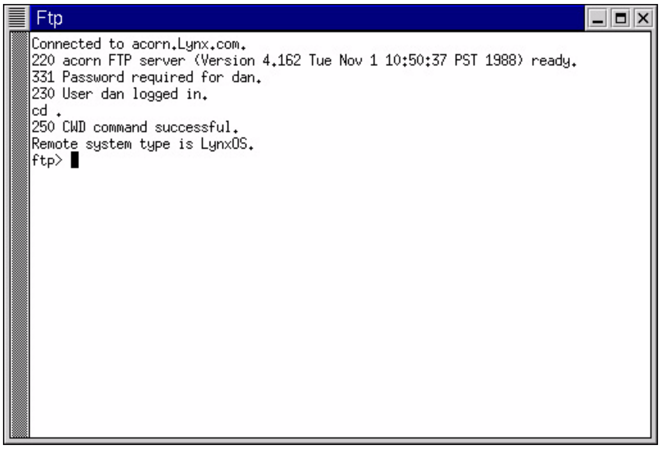
Click on the **Ftp** button to open an FTP connection with the target. The auto-login facility of `ftp` is used. After the FTP connection is established, files can be copied to and from the host and target.

To configure **Ftp**, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Password
- FTP Port
- FTP Target Directory

NOTE: If the FTP Target Directory is empty in the **Target System** panel, the Target System Directory field is used.

The following figure illustrates a sample FTP session activated by clicking the **Ftp** button:



```
Connected to acorn.Lynx.com.
220 acorn FTP server (Version 4.162 Tue Nov 1 10:50:37 PST 1988) ready.
331 Password required for dan.
230 User dan logged in.
cd .
250 CWD command successful.
Remote system type is LynxOS.
ftp> █
```

Figure 3-3: FTP Session

Target Process Viewer

Provides a view of target system activity.

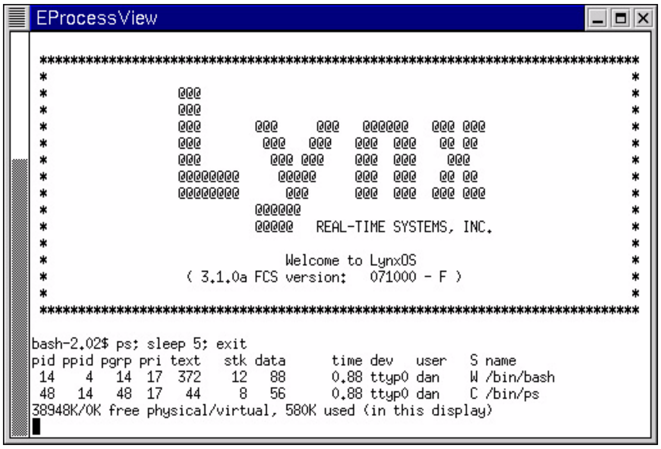
Click on the **Target Process Viewer** button to open a telnet session to the remote target and execute the command specified by the **Process Viewer Command**.

To configure the **Target Process Viewer**, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Password
- Process Viewer Command

Any target-supported process view command (`ps`, for example) can be specified as the **Process Viewer Command**.

The following figure shows the result of clicking the **Target Process Viewer** button.


EProcessView

```
******************************************************************************
*                                                                            *
*                @@@                                                         *
*                @@@                                                         *
*                @@@       @@@     @@@   @@@@@@   @@@ @@@                    *
*                @@@        @@@   @@@   @@@  @@@   @@ @@                     *
*                @@@         @@@ @@@    @@@  @@@    @@@                      *
*                @@@@@@@@     @@@@@     @@@  @@@   @@ @@                     *
*                @@@@@@@@      @@@      @@@  @@@  @@@ @@@                    *
*                          @@@@@@                                            *
*                          @@@@@   REAL-TIME SYSTEMS, INC.                   *
*                                                                            *
*                              Welcome to LynxOS                             *
*                 ( 3.1.0a FCS version:   071000 - F )                       *
*                                                                            *
******************************************************************************

bash-2.02$ ps; sleep 5; exit
pid ppid pgrp pri text   stk data      time dev   user   S name
 14    4   14  17  372    12   88      0.88 ttyp0 dan    W /bin/bash
 48   14   48  17   44     8   56      0.88 ttyp0 dan    C /bin/ps
38948K/0K free physical/virtual, 580K used (in this display)
```

Figure 3-4: Target Process Viewer

System Viewer

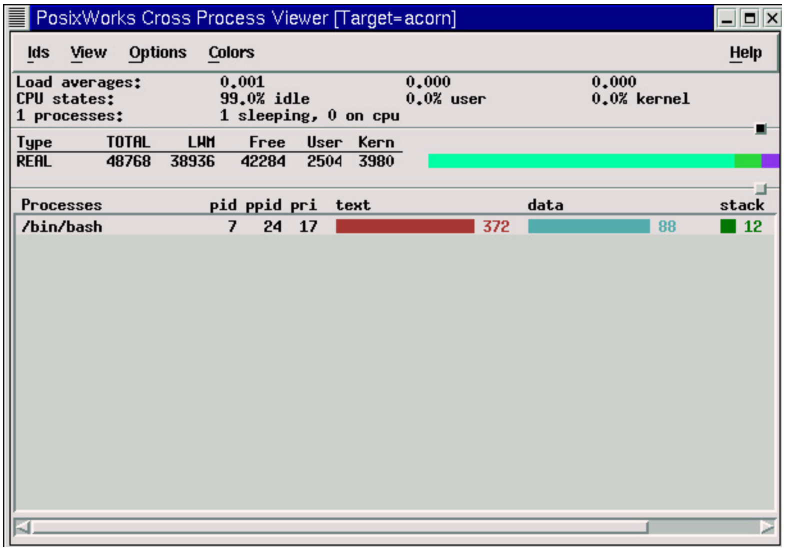
Views target system activity.

Click on the **System Viewer** button to launch the LynxOS cross-process viewer server on the target and the LynxOS cross-process viewer client on the host. A real-time histogram detailing target system activity is displayed.

To configure **System Viewer**, set the following field in the **Target System** panel:

- LAN Address
- User name

The following figure shows a sample **System Viewer** window.



```
Load averages:          0.001           0.000           0.000
CPU states:             99.0% idle      0.0% user       0.0% kernel
1 processes:            1 sleeping, 0 on cpu

Type    TOTAL   LWM    Free   User  Kern
REAL    48768   38936  42284  2504  3980

Processes            pid ppid pri  text        data      stack
/bin/bash              7   24  17       372         88      12
```

Figure 3-5: The System Viewer Window

Build and Copy Executable

Downloads the current project executable to the target system.

Click the **Build and Copy Executable** button to download the current project executable on the target system. Before starting the operation, the modification time of the sources is compared with the modification time of the current project executable (if any) and, if necessary, the executable is rebuilt.

To configure **Build and Copy Executable**, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Target Directory

To provide remote access, the `.rhosts` file in the user's home directory on the target system must be modified to allow access from the host by adding the host system name and user name. See "Accessing the Target System" on page 20 for more information.

Copy Executable

Copies the current project executable onto the target system.

Click on the **Copy Executable** button to copy the current project executable to **Target Directory** without checking the modification time of the sources and the executable.

To configure **Copy Executable**, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Target Directory

To provide remote access, the `.rhosts` file in the user's home directory on the target system must be modified to allow access from the host. See "Accessing the Target System" on page 20 for more information.

Target Command Execution

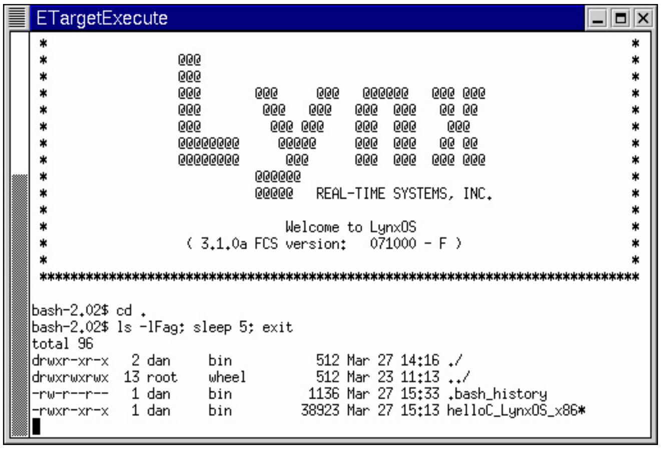
Executes a command on the target system.

Click the **Target Command Execution** button to open a `telnet` session on the remote target and execute the command specified by the **Target Execution Command**. The result of the command is displayed on the host in an xterm window.

To configure **Target Command Execution**, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Password
- Target Execution Command

The following figure shows the result of the **`ls -la`** command using the **Target Command Execution** button.



```
 *                                                                            *
 *                 @@@                                                        *
 *                 @@@                                                        *
 *                 @@@       @@@     @@@   @@@@@@   @@@ @@@                   *
 *                 @@@        @@@   @@@   @@@  @@@   @@ @@                    *
 *                 @@@         @@@ @@@    @@@  @@@    @@@                     *
 *                 @@@@@@@@     @@@@@     @@@  @@@   @@ @@                    *
 *                 @@@@@@@@      @@@      @@@  @@@  @@@ @@@                   *
 *                           @@@@@@                                           *
 *                           @@@@@   REAL-TIME SYSTEMS, INC.                  *
 *                                                                            *
 *                               Welcome to LynxOS                            *
 *                  ( 3.1.0a FCS version:   071000 - F )                      *
 *                                                                            *
 ******************************************************************************

bash-2.02$ cd .
bash-2.02$ ls -lFag; sleep 5; exit
total 96
drwxr-xr-x   2 dan     bin           512 Mar 27 14:16 ./
drwxrwxrwx  13 root    wheel         512 Mar 23 11:13 ../
-rw-r--r--   1 dan     bin          1136 Mar 27 15:33 .bash_history
-rwxr-xr-x   1 dan     bin         38923 Mar 27 15:13 helloC_LynxOS_x86*
█
```

Figure 3-6: Target Command Execution

GDB

Starts a GDB session.

Click the **GDB** button to start `gdbserver` on the target, execute `gdb` on the host, and establish a connection between `gdbserver` and `gdb`. Two xterm windows appear on the host:

- The `gdbserver` window (for gdbserver running on the target)
- The `gdb` window (for GDB running on the host)

Before starting a GDB session, the modification time of the sources is compared with the modification time of the current executable. If necessary, the executable is rebuilt. After the executable is rebuilt, it is downloaded to the target.

The `gdb` window is used to enter GDB commands. The `gdbserver` window is used to restart gdbserver after it terminates.

To restart the GDB session:

1. Restart gdbserver in the `gdbserver` window by typing:

   ```
   # gdbserver host:port executable_name
   ```

   where *host* and *port* correspond to the LAN Address and Port Number fields of the **Target System** panel, and *executable_name* is the name of the application.

2. Connect with `gdbserver` from the `gdb` window by typing:

   ```
   # target remote host:port
   ```

   where *host* and *port* correspond to the LAN Address and Port Number fields of the **Target System** panel.

On successful completion of these steps, the connection between `gdb` and `gdbserver` is restored.

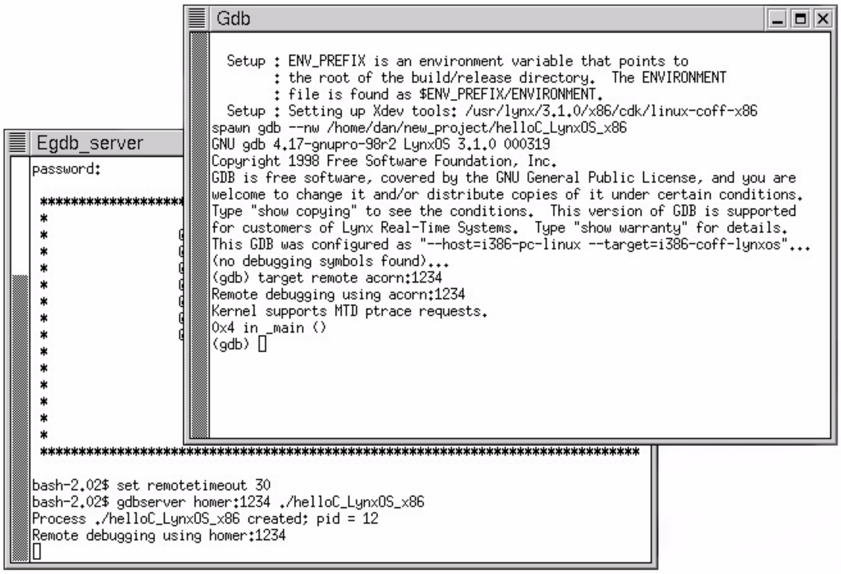
To configure GDB, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Password
- Type of GDB Connection (LAN or Serial)
- Port Number (for LAN connections)
- Serial Port (for Serial connections)
- Target Directory

Close the `gdbserver` and `gdb` windows to stop the GDB session.

NOTE: The default name of the serial device on the Linux host is `/dev/ttyS0`. To change this setting, copy the `/etc/.CWLynuxWorks` file to the home directory and edit the string corresponding to the command used to invoke GDB in this file. Refer to Chapter 5, "Customizing the Interface" for details.

The following figure illustrates a sample GDB session:



```
  Setup : ENV_PREFIX is an environment variable that points to
        : the root of the build/release directory.  The ENVIRONMENT
        : file is found as $ENV_PREFIX/ENVIRONMENT.
  Setup : Setting up Xdev tools: /usr/lynx/3.1.0/x86/cdk/linux-coff-x86
spawn gdb --nw /home/dan/new_project/helloC_LynxOS_x86
GNU gdb 4.17-gnupro-98r2 LynxOS 3.1.0 000319
Copyright 1998 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.  This version of GDB is supported
for customers of Lynx Real-Time Systems.  Type "show warranty" for details.
This GDB was configured as "--host=i386-pc-linux --target=i386-coff-lynxos"...
(no debugging symbols found)...
(gdb) target remote acorn:1234
Remote debugging using acorn:1234
Kernel supports MTD ptrace requests.
0x4 in _main ()
(gdb) 
```

```
password:

 ******************
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 ****************************************************************************

bash-2.02$ set remotetimeout 30
bash-2.02$ gdbserver homer:1234 ./helloC_LynxOS_x86
Process ./helloC_LynxOS_x86 created; pid = 12
Remote debugging using homer:1234
```

Figure 3-7: A Sample GDB Session

TotalDB

Starts a GDB session with a graphical interface on the host.

Click the **TotalDB** button to start `gdbserver` on the remote target, launch the graphical interface of the debugger on the host, and establish a connection between the two.

Before calling Total/db, the modification times of the source files are compared with the modification time of the current executable. If necessary, the executable is rebuilt. After the executable is rebuilt, it is downloaded to the target.

Two xterm windows appear on the host:

- The `gdbserver` window (for gdbserver running on the target)
- The `TotalDB` window (graphical interface to GDB)

To start a debug session, click the **Continue** button from the **Control** menu.

To restart a GDB session:

1. Restart `gdbserver` with this command:

   ```
   # gdbserver host:port executable_name
   ```

   where *`host`* and *`port`* correspond to the LAN Address and Port Number fields of the **Target System** panel, respectively.

2. Click the **Connect to Target** button from the **Run** menu in the **TotalDB** window.

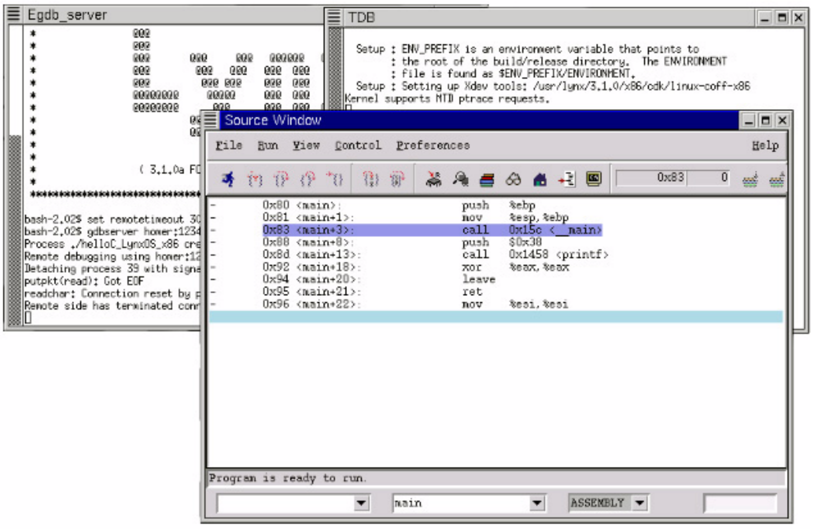
Upon successful completion of these steps, the connection between gdbserver and Total/db is restored.

To configure **TotalDB**, set the following fields in the **Target System** panel:

- LAN Address
- User Name
- Password
- Type of GDB Connection (LAN or Serial)
- Port Number (for LAN connections)
- Serial Port (for Serial connections)
- Target Directory

NOTE: The default name of the serial device on the Linux host is `/dev/ttyS0`. To change this setting, copy the `/etc/.CWLynuxWorks` file to the home directory and edit the string corresponding to the command used to invoke Total/db. Refer to Chapter 5, "Customizing the Interface" for details.

The following figure illustrates a sample Total/db session:



```
bash-2.02$ set remotetimeout 30
bash-2.02$ gdbserver homer:1234
Process ./helloC_LynxOS_x86 cre
Remote debugging using homer:12
Detaching process 39 with signa
putpkt(read): Got EOF
readchar: Connection reset by p
Remote side has terminated conn
```

```
 Setup : ENV_PREFIX is an environment variable that points to
       : the root of the build/release directory.  The ENVIRONMENT
       : file is found as $ENV_PREFIX/ENVIRONMENT.
 Setup : Setting up Xdev tools: /usr/lynx/3.1.0/x86/cdk/linux-coff-x86
Kernel supports MTD ptrace requests.
```

```
-        0x80 <main>:            push   %ebp
-        0x81 <main+1>:          mov    %esp,%ebp
-        0x83 <main+3>:          call   0x15c <__main>
-        0x88 <main+8>:          push   $0x38
-        0x8d <main+13>:         call   0x1458 <printf>
-        0x92 <main+18>:         xor    %eax,%eax
-        0x94 <main+20>:         leave
-        0x95 <main+21>:         ret
-        0x96 <main+22>:         mov    %esi,%esi

Program is ready to run.
```

Figure 3-8: A Sample Total/db Session

Host Shell

Invokes a shell on the host system.

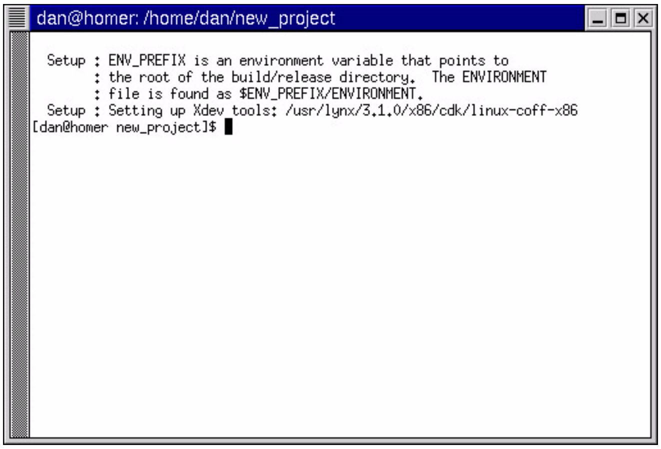
Click the **Host Shell** button to start a shell with the necessary settings (including environment variables) on the host system in the current project directory. **Host Shell** can be used to access the appropriate host GNU tools or execute other actions on the host.

To configure **Host Shell**, set the following field in the **Target System** panel:

- Compiler Prefix Path

NOTE: If the Compiler Prefix Path field is empty, the directory specified in the cross-development tools location file `cdt.cfg` is used.

The following figure shows the **Host Shell**:



```
  Setup : ENV_PREFIX is an environment variable that points to
        : the root of the build/release directory.  The ENVIRONMENT
        : file is found as $ENV_PREFIX/ENVIRONMENT.
  Setup : Setting up Xdev tools: /usr/lynx/3.1.0/x86/cdk/linux-coff-x86
[dan@homer new_project]$ 
```

Figure 3-9: The Host Shell Window

Help

Opens the online Help system.

Click the **Help** button to open the HTML-formatted *LynuxWorks Development Tools, CodeWarrior IDE Edition User's Guide* and LynuxWorks man pages.

Target System Panel

The **Target System** panel allows the user to enter configuration information for the remote target system. The following table describes the **Target System** panel fields:

Table 3-1: Target System Panel Fields

Field	Description	Default Setting
Project Name	Displays the name of the current project. (Cannot be changed by the user.)	Current project name
Target Directory	Changes the directory on the target system used to store downloaded or copied files.	Target user's home directory
LAN Address	Sets the IP address or host name of the target system.	127.0.0.1 (current host)
User Name	Sets the user account used to perform actions on the target system.	Current user account on the host
Password	Sets the user password on the target. This password is used to authenticate telnet and FTP connections.	Empty string
Target Execution Command	Sets the command to be executed on the target.	`ls -lFag; sleep 5; exit`
Process Viewer Command	Sets the command to be executed on the target.	`ps; sleep 5; exit`
GDB Connection Type	Sets connection type with the GDB server on the target. Two connection types are available: - LAN (over Ethernet using TCP/IP) - Serial Port	LAN
GDB Port	Sets the port number used to connect to the GDB server on the target. This option is available only if the `LAN connection` option is selected in the **GDB Connection Type** field.	1234

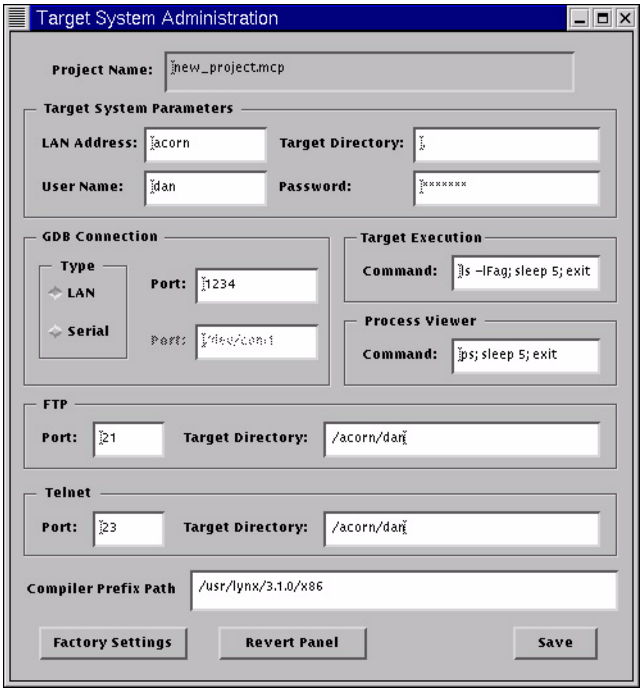
Table 3-1: Target System Panel Fields (Continued)

Field	Description	Default Setting
GDB Serial Port	Sets the serial device on the target used to connect to the GDB. This option is available only if the `Serial Port connection` option is selected in the **GDB Connection Type** field.	`/dev/com1`
FTP Port	Sets the port number used to establish an FTP connection.	21
FTP Target Directory	Sets the name of the directory on the target.	Empty string
Telnet Port	Sets the number of the port used to establish a telnet connection.	23
Telnet Target Directory	Sets the name of the telnet home directory on the target.	Empty string
Compiler Prefix Path	Changes the path to the cross-development tools top directory specified in the cross-development tools location file `cdt.cfg`.	Empty string

NOTE: When setting the Compiler Prefix Path, ensure that the path is correct on the target. This path is not verified by the system, and a wrong path can result in build errors.

Projects can contain a number of build targets for different remote systems. The **Target System** panel contents may vary.

The following figure illustrates the **Target System** panel:



Project Name: new_project.mcp

Target System Parameters

LAN Address: acorn
Target Directory:
User Name: dan
Password: *******

GDB Connection

Type: LAN / Serial
Port: 1234
Port: /dev/com1

Target Execution

Command: ls -lFag; sleep 5; exit

Process Viewer

Command: ps; sleep 5; exit

FTP

Port: 21
Target Directory: /acorn/dan

Telnet

Port: 23
Target Directory: /acorn/dan

Compiler Prefix Path: /usr/lynx/3.1.0/x86

Figure 3-10: Target System Panel

Current settings for the target are saved in the `.CodeWarrior/LynuxWorks/.target` file in the user's home directory.

CHAPTER 4

Target System Configurations

This chapter describes the supported LynxOS/BlueCat Linux target system configurations and options.

LynxOS Target Systems

The LynuxWorks Development Tools, CodeWarrior IDE Edition product works with both hard-disk and RAM-disk based file systems on the target. The target kernel must be configured to support the following requirements:

- TCP/IP
- Sufficient `pty` devices
- Serial driver (for serial port connections)

Refer to the *LynxOS User's Guide* for detailed descriptions of the kernel configuration process.

Hard Disk Targets

The standard LynxOS installation includes the required configurations for cross-development. Refer to the *LynxOS Installation Guide* for instructions on installing LynxOS on a hard disk.

To support the features of the **LynuxWorks** menu, the following target files must be modified after the installation:

- The `/net/rc.network` file must set the target system name and launch the `inetd` daemon to allow network connections.
- The `/etc/inetd.conf` and `/etc/services` files must be modified to enable telnet, FTP, and remote shell connectivity.

- The `/etc/hosts` file must contain the host name and IP address of the development host where LynuxWorks Development Tools, CodeWarrior IDE Edition is running.
- The `.rhosts` file, located in the home directory of the user, must include the development system host name, as well as a user name. Additionally, the `/etc/passwd` file can be modified to allow unrestricted logins.

Diskless Targets

To simplify working with RAM-based file systems, LynuxWorks Development Tools, CodeWarrior IDE Edition includes a sample specification file, `cw.spec`. This sample specification contains instructions on generating a ROM-able kernel and the root file system. The root file system created using this specification file contains all utilities and files necessary for cross development.

The sample specification file, `cw.spec`, is located in the `/usr/local/metrowerks/CodeWarrior_Pro6/LynuxWorks \ /system/LynxOS` directory, along with the target-specific files that must be modified before making the kernel image with **`mkimage`**. The following table provides a list of modifications required to these files:

Table 4-1: Files/Modification List

File	Modification
`rc.network`	Specify the host name and IP address of the target, as well as an appropriate network device.
`passwd`	Specify the user account. All actions on the target are performed as this user account.
`hosts`	Specify the host name and IP address of the development host and target.
`.rhosts`	Specify the development system's hostname and the user name on the host.
`cw.spec`	Specify the macros appropriate to the target system.
`build.sh`	KDI build script
`rc`	Runs `rc.network`.
`ttys-386`	Sets terminal defaults.

Table 4-1: Files/Modification List (Continued)

File	Modification
`ttys-mips`	Sets terminal defaults.
`ttys-ppc`	Sets terminal defaults.

After the kernel image is built with `mkimage`, it can be booted on the target. Refer to the *LynxOS User's Guide* for a detailed description of `mkimage` and the KDI boot process.

Generating a KDI

The `build.sh` script is used to generate a kernel downloadable image (KDI). The `build.sh` script prompts the user for a BSP name, processes parameters from the `cw.spec` file, and builds a kernel image.

To build a KDI with the `build.sh` script, use the following instructions:

1. Copy all files from `/usr/local/metrowerks/CodeWarrior_Pro6/\ Lynuxworks/system/LynxOS` to a directory on the hard disk. For example, the following command copies the files to the `$HOME/CW_kdi` directory:

   ```
   # cp -R /usr/local/metrowerks/CodeWarrior_Pro6/\
   LynuxWorks/system/LynxOS $HOME/CW_kdi
   ```

2. Set the LynxOS cross-development environment. For example, to set the bash environment:

   ```
   # cd /usr/lynx/3.1.0a/mips
   # . SETUP.bash
   ```

3. Change to the directory on the hard disk where the files copied in Step 1 are stored. For example,

   ```
   # cd $HOME/CW_kdi
   ```

4. Edit the `rc.network`, `hosts`, and `.rhosts` files. Refer to Table 4-1, “Files/Modification List,” on page 36 for a description of the required modifications.

5. Run the `build.sh` script:

   ```
   # ./build.sh
   ```

   Type a target BSP name when prompted and press **Enter**.

As a result of this procedure, the `cw.kdi` file is built. The resulting file system includes the user account `test` with the password `test_123`.

BlueCat Linux Target Systems

The LynuxWorks Development Tools, CodeWarrior IDE Edition product works with both hard-disk and RAM-disk based file systems on the target. The target kernel must be configured with the following parameters enabled:

- Networking support (`CONFIG_NET`)
- TCP/IP networking support (`CONFIG_INET`)
- Network device support (`CONFIG_NETDEVICES`)
- Ethernet card support
- RAM disk support (`CONFIG_BLKDEV_RAM`)
- BlueCat Linux RFS support (`CONFIG_BLUECAT_RFS`)
- UNIX 98 PTY support (`CONFIG_UNIX98_PTY`, `CONFIG_DEVPTS_FS`) with sufficient `pty` devices
- Serial driver support (for communication via the serial port)
- IPC support (`CONFIG_SYSV_IPC` for IPC using stationaries)

Refer to the *BlueCat Linux User's Guide* for a detailed description of the kernel configuration process.

Diskless Targets

To simplify working with RAM-based file systems, LynuxWorks Development Tools, CodeWarrior IDE Edition includes a sample specification file, `cw.spec`. This sample specification file contains instructions required to generate the target root file system. The root file system created using this specification file contains all utilities and files necessary for cross development using LynuxWorks Development Tools, CodeWarrior IDE Edition.

The sample specification file, `cw.spec`, is located in the `/usr/local/metrowerks/CodeWarrior_Pro6/LynuxWorks\ /system/BlueCat` directory, along with the target-specific files located in the local subdirectory. These files must be modified before making the root file system with the **`mkrootfs`** utility. The following table provides a list of modifications required to these files:

Table 4-2: Files/Modification List

File	Modification
`rc.sysinit`	Specify the host name and IP address of the target, as well as an appropriate network device.
`shadow`	Specify the password of the user on the target.
`passwd`	Specify the user account. All actions on the target are performed as this user account.
`hosts`	Specify the host name and IP address of the development host and target.
`.rhosts`	Specify the development system's host name and the user name on the host.
`cw.spec`	Specify the `USER_ID` variable.

After the kernel and the root file system are built with `mkrootfs` and `mkkernel`, the kernel can be booted on the target. Refer to the *BlueCat Linux User's Guide* for a detailed description of the boot process.

Generating BlueCat Target Images

The `build.sh` script is used to generate BlueCat downloadable images.

To build the images with the `build.sh` script, the user must do the following:

1. Copy all files from `/usr/local/metrowerks/CodeWarrior_Pro6/\ LynuxWorks/system/BlueCat` to an empty directory on the development host. For example, the following command copies the files to the `$HOME/CW_kdi` directory:

   ```
   # cp -R /usr/local/metrowerks/CodeWarrior_Pro6/\
   LynuxWorks/system/BlueCat $HOME/CW_kdi
   ```

2. Set the BlueCat Linux cross development environment. For example, to set the bash environment:

   ```
   # cd /home/BlueCat_arm
   # . SETUP.sh
   ```

3. Change to the directory on host where the files copied in Step 1 are stored. For example,

   ```
   # cd $HOME/CW_kdi
   ```

4. Edit the `rc.sysinit`, `hosts`, `passwd`, `shadow`, and `.rhosts` files. Refer to Table 4-2, “Files/Modification List,” on page 39 for a description of the required modifications.

5. Run the `build.sh` script:

   ```
   # ./build.sh
   ```

As a result of this procedure, the `cw.rfs`, `cw.kernel`, and `cw.kdi` files are built. The resulting file system includes the user account `test` with the password `test_123`. These images can now be booted onto the BlueCat Linux target. Refer to the *BlueCat Linux User’s Guide* for details.

Installing BlueCat Linux on a Target Disk

For more information about installing a BlueCat Linux kernel and root file system on a target hard disk, refer to “Booting BlueCat Linux from Hard Disk” in the *BlueCat Linux User’s Guide*.

CHAPTER 5

Customizing the Interface

This chapter describes how to customize the LynuxWorks Development Tools, CodeWarrior IDE Edition. Configurable options include:

- The Standard toolbar
- Location of the cross-development tools
- The default actions of the **LynuxWorks** menu buttons

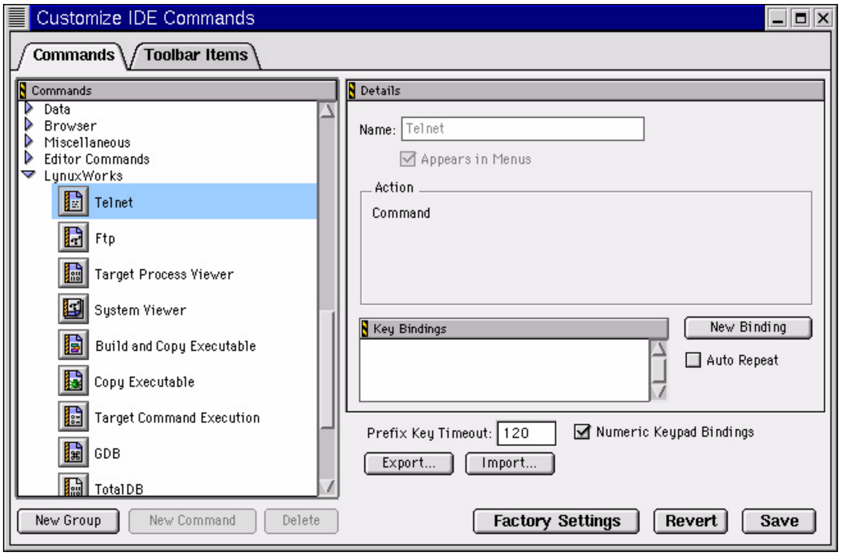
Adding LynuxWorks Buttons to the Toolbar

To add a LynuxWorks specific button to the standard menu bar:

1. Click the **Command & Key Bindings** button in the **Edit** menu.
2. Open the **LynuxWorks** folder.
3. Select the LynuxWorks item to add.
4. Drag-and-drop the selected icon onto the toolbar.
5. Close the **Command & Key Bindings** window.

The new icon appears on the **Project** toolbar.

The following figure illustrates the **Customize IDE Commands** window:



Name: Telnet

☑ Appears in Menus

Command

☐ Auto Repeat

Prefix Key Timeout: 120

☑ Numeric Keypad Bindings

Figure 5-1: Customizing Commands Window

Setting the Cross-Development Tools Location

Location of the cross-development tools is specified during installation of LynuxWorks Development Tools, CodeWarrior IDE Edition. The location is saved in the `cdt.cfg` file of the `/usr/local/metrowerks \ /CodeWarrior_Pro6/LynuxWorks` directory. This file contains a special string for each cross-development environment variable on the host. The string uses the following format:

```
OS     platform  directory
```

where:

- *`OS`* - The name of target operating system (LynxOS or BlueCat Linux)
- *`platform`* - Target architecture:
  - `x86`, `ppc`, or `mips` for LynxOS
  - `x86`, `ppc`, `arm`, or `sh` for BlueCat Linux
- *`directory`* - The cross-development tools top-level directory

  This directory name is used to set the **Compiler Prefix Path** field for the **LynuxWorks** menu button.

Modify the `cdt.cfg` file if the location of the cross-development environment changes after installation of the LynuxWorks CodeWarrior IDE Edition.

Changing LynuxWorks Menu Button Actions

The `.CWLynuxWorks` file in the user's home directory defines the commands that are executed when clicking a menu button. If this file is not present in this directory, the system file `/etc/.CWLynuxWorks` is used.

The `.CWLynuxWorks` file uses name definitions that correspond to the button names of the **LynuxWorks** menu. The body of each section defines the command that runs when a button is clicked. The command string contains macros that use settings from the **Target System** panel.

For example, the following defines the **System Viewer** button:

```
[File Download]
```

```
SVIEW = /usr/local/metrowerks/CodeWarrior_Prog/\
LynuxWorks/bin/SystemViewer %a %n
```

Clicking the **System Viewer** button in this example starts the System Viewer command with two arguments: the target system IP address (`%a`) and the user account on the target (`%n`).

Supported macros are described in the `/etc/.CWLynuxWorks` file.

Index