Installation Guide

version 4.9

Peter Högfeldt

1997-11-06

Typeset in LaTeX from SGML source using the DOCBUILDER 3.0 Document System.

Contents

Chapter 1

Installation Guide

This chapter is about installing the Development Environment on UNIX or Windows NT.

1.1 Introduction

Introduction

This *Installation Guide* is intended for

- UNIX system administrators who install and maintain the *Erlang* system,
- users installing *Erlang* on Windows NT.

Note however, that for an ordinary installation on UNIX no special privileges are necessarily required; i.e. an end user may act as administrator, if appropriate.

In this guide you will find information on:

- Installation of Development Environment
- Verification of Installation
- Installation of Embedded Environment
- Starting an Embedded Environment
- Corrective Maintenance
- Preparing and sending Problem Reports on the system.

Prerequisites

For UNIX systems, the installation parts of this guide assume that the user is familiar with how to install software on a UNIX system.

Typographic Conventions

System element names, commands, and other system components are shown in `non-proportional font`, e.g. “the directory `/usr/local`”.

1.2 Installation of Development Environment

This chapter is about installing the Development Environment on UNIX or Windows NT. A Development Environment consists of all functionality needed for development, e.g. run-time system, compiler, standard libraries, graphics package, full documentation sets, but no Embedded Environment specific functionality as automatic start at reboot, activation of hardware watchdog etc.

UNIX in General

The Erlang system runs on a number of different UNIX systems. The installation is performed in a similar way for all of them. Refer to the individual subsections below for specific hardware and software requirements and any details that may differ.

Hardware and Software Requirements

The system is delivered as a single compressed tar file, and the installation requires approximately 50 MB of file system space, including the printed documentation in machine readable format. This can, if desired, be removed when the installation is completed.

As UNIX supports virtual memory, no strict minimum of physical memory is required, but as with all applications, shortage of physical memory will have negative effects on performance. The amount of virtual memory used by the Erlang system is mostly dependent on the amount of code loaded, and the number and complexity of Erlang processes running. Eight to ten MB is not unusual for a fairly complex system.

To browse the on-line HTML documentation, Netscape 3.0 or equivalent browser is needed.

Installation Procedure

When installed, the entire system, except for a small start-up script, resides in a single directory tree. The location of this directory tree can be chosen arbitrarily by the installer, and it does not need to be in the user's `$PATH`. The only requirements are that the file system where it is placed has enough free space (see above), and that the users who run the Erlang system have read access to it. In the example below, the directory tree is assumed to be located at `/usr/local/erlang`, which is here called the *top-level directory*.

It is assumed that you have got a compressed tape archive file (tar file), the name of which is `<PREFIX>.tar.Z`, or `<PREFIX>.tar.gz`, where `<PREFIX>` is a string denoting the particular Development Environment release, e.g. `otp_LXA11928_sunos5_R2A`.

Wherever the string `<PREFIX>` is used below, it should be replaced by the actual name prefix of the compressed tar file.

The tape archive file does not have one single directory in which all other files are rooted. Therefore the tape archive file must be extracted into an empty (newly created) directory.

1. If the *top-level directory* does not already exist, create it:

```
mkdir /usr/local/erlang
```

2. Change the current directory to the *top level directory*:

```
cd /usr/local/erlang
```

3. Create the *installation directory* in the current directory. A suitable name is <PREFIX>, e.g.

```
mkdir /usr/local/erlang/otp_LXA11928_sunos5_R2A
```

4. Change to the *installation directory*, e.g.

```
cd otp_LXA11928_sunos5_R2A
```

5. The compressed tar file is assumed to reside in the directory <SOME-DIR>. Extract the compressed tar file into the current directory by using *precisely one* of the following commands (depending on wether the suffix of the file name is .Z or .gz):

```
zcat <SOME-DIR>/<PREFIX>.tar.Z | tar xfp -
gunzip -c <SOME-DIR>/<PREFIX>.tar.gz | tar xfp -
```

6. Read the README file in the installation directory for last minute updates, before proceeding.
7. Run the Install script in the installation directory, with the absolute path of the installation directory as argument,

```
./Install /usr/local/erlang/<PREFIX>
```

   and supply answers to the prompts.
   In most cases, there is a default answer in square brackets ([]). If the default is satisfactory, just press <Return>. In general you are only prompted for two pieces of information:

   - The default boot script. For a plain development environment you should use start_clean, but if you are for instance developing for an embedded system, start_sasl is appropriate.
   - For those systems where the X Window interface pxw is included, the version and location of the X libraries. Pxw does not require any X libraries in the system, but using existing libraries will normally result in a smaller binary file. However, if existing libraries are to be used, they must be compatible with the MIT X11R5 release (X11 R6 is).

   The script will in some cases link object files in the distribution into executable binaries for the current system.
8. Make the Development Environment available for users, either by putting the path /usr/local/erlang/<PREFIX>/bin in users $PATH variable, or link the executable /usr/local/erlang/<PREFIX>/bin/erl accordingly, e.g.:

```
ln -s /usr/local/erlang/<PREFIX>/bin/erl /usr/local/bin/erl
```

On UNIX systems where there is a choice between different methods of name resolution, the user can control his choice by setting the environment variable ERL_RESOLV to a suitable value. If ERL_RESOLV is

set to `nodns`, the configuration of the underlying UNIX system defines the name resolution scheme to use. If `ERL_RESOLV` is set to `dns`, or if `ERL_RESOLV` is not defined, only the resolver of the Domain Name System (DNS) is used.

Which name resolution method to choose depends on how the computer(s) in your network have been configured for name resolution. If in doubt, consult your local system administrator.

SunOS 4

Hardware and Software Requirements

The Erlang system for `SunOS4` runs on any Sun4 system. Due to the way the installation is done, it should be possible to run an Erlang system on any release of SunOS 4. However, there are many bugs in the early versions, and it is recommended that at least SunOS 4.1.4 is used.

Installation Procedure

No version of Sun's OpenWindows for SunOS 4 based on X11R5 has been released, and thus the OpenWindows libraries are not usable by `pxw`. If the MIT X11R5 release has been installed, pxw can use those libraries, otherwise just answer 'no' to the `Install` script's question about X11R5-compatible libraries.

SunOS 5 (Solaris 2)

Hardware and Software Requirements

Like the SunOS 4 version, this version will run on any Sun 4 system. The minimum OS version recommended is 5.5.1, i.e. Solaris 2.5.1.

If you are installing an embedded system, patch 103640-02 should be added.

Installation Procedure

The X libraries in OpenWindows release 3.2 (which is the one included in Solaris 2.5.1) are based on X11R5, and is thus usable by pxw. However, as no C compiler is bundled with Solaris 2, all executable files are pre-linked. `pxw` is linked to expect the X libraries in `/usr/openwin/lib`.

Linux

Hardware and Software Requirements

Erlang system for Linux runs on x86 Personal Computers. A minimum of 16MB memory and swapping-enabled is recommended.

Installation Procedure

The installation procedure does not differ from the description for installation on UNIX in general.

Windows NT

Hardware and Software Requirements

Erlang system for Windows NT runs on x86 Personal Computers. A minimum of 16MB memory is recommended.

The minimum OS version required is Windows NT 4.0. If distributed Erlang, or the Erlang networking applications `socket` or `udp`, or any of the graphics tools based on `gs` are to be used, networking must be configured and Windows Sockets version 2.0 (which is a standard component of Windows NT 4.0) must be available on the system.

To browse the on-line HTML documentation, Microsoft Internet Explorer 3.0, Netscape 3.0 or equivalent browser is needed.

Installation Procedure

The installation procedure is is automated, except for a few initial and final steps.

It is assumed that you have got an Erlang distribution in one of the following forms:

- a file named <PREFIX>.zip, or
- a file named <PREFIX>.tar, or
- A CD-ROM containing the Erlang distribution.

Here <PREFIX> is a string denoting the particular release version, e.g. otp_LXA11928_win40_R2A. Whenever the string <PREFIX> occurs below, it should during the installation procedure be replaced by the actual name.

Distribution is on file <PREFIX>.zip

1. Move the file <PREFIX>.zip to a temporary directory, e.g. the directory

   ```
   D:\TMP
   ```

2. Unzip the contents of the file (use for instance the *WinZip* application).
3. Use *Windows Explorer* for navigating to the directory where the distribution was unzipped.
4. Read the README file in that directory (double-click on the icon) for last minute updates before proceeding, and follow the instructions in that file, if there are any.
5. In the same directory, run the SETUP.EXE program which will automatically install the complete distribution.
6. Go to *Final Installation Step* below.

Distribution is on file <PREFIX>.tar

1. Move the file <PREFIX>.tar to a temporary directory, e.g.

   ```
   D:\TMP
   ```

2. Next you have to unpack the contents of the file. If you have the *WinZip* application installed, use it for unpacking the file (it works for both .zip and .tar files).
   If you do not have the *WinZip* application, hold of a copy of tar.exe for Windows NT, and extract the file in the temporary directory chosen, as follows,

   ```
   tar xvf <PREFIX>.tar
   ```

3. Use *Windows Explorer* for navigating to the directory where the distribution was extracted.
4. Read the README file in that directory (double-click on the icon) for last minute updates before proceeding, and follow the instructions in that file, if there are any.
5. In the same directory run the SETUP.EXE program, which will automatically install the complete distribution.
6. Go to *Final Installation Step* below.

Distribution is on CD-ROM

1. Use *Windows Explorer* for navigating to the directory on the CD-ROM where the Erlang distribution is located.
2. Read the README file in that directory (double-click on the icon) for last minute updates before proceeding, and follow the instructions in that file, if there are any.
3. In the same directory, run the SETUP.EXE program which will automatically install the complete distribution.
4. Go to *Final Installation Step* below.

Final Installation Steps

1. When the installation has ended successfully, a shortcut Erlang icon has been created on your desktop, and the initial current directory for Erlang will be set according to the environment variables HOMEDRIVE and HOMEPATH, i.e. the users home directory.
   If the initial current directory set by the installation is not what you want, reconfigure the shortcut accordingly.

Erlang can now be started by double-clicking on the shortcut icon.

1.3 Installation Verification

This chapter is about verifying your installation by performing a few simple tests to see that your system is properly installed.

If any of the tests listed below fail, consult the chapter on *Corrective Maintenance*.

Installation verification on UNIX

- Start Erlang from the command line,

```
unix> erl
```

  Expect the following output:

```
Erlang (BEAM) emulator version 4.9

Eshell V4.9  (abort with ^G)
1>
```

  Exit by entering the command `halt()`,

```
1> halt().
unix>
```

  Note: The trailing full stop (`"."`) is an end marker for all commands in the Erlang shell, including the command `halt()`, and must be entered for a command to begin execution.

- Start the Integrated Development Environment from the command line,

```
erl -x
```

  and check that the main window of the pxw-based Integrated Development Environment pops up. Exit the Integrated Development Environment by clicking the `File/Quit` menu.

- Start the gs-based toolbar from the Erlang shell (do not forget to end the command with a full stop as shown),

```
unix> erl
Erlang (BEAM) emulator version 4.9

Eshell V4.9  (abort with ^G)
1> toolbar:start().
```

and check that the toolbar window pops up. Exit by clicking the `File/Quit` menu. Exit Erlang by entering the `halt()` command at the Erlang shell prompt as above.

Installation verification on Windows NT

- Start Erlang by double-clicking on the Erlang shortcut icon on the desktop.
  Expect a command line window to pop up with the following output,

```
Erlang (BEAM) emulator version 4.9

Eshell V4.9  (abort with ^G)
1>
```

- Start the gs-based toolbar from the Erlang shell,

```
1> toolbar:start().
2>
```

  and check that the toolbar window pops up.
  Note: The trailing full stop (`"."`) is an end marker for all commands in the Erlang shell, and must be entered for a command to begin execution.
- Exit by entering the command `halt()`,

```
1> halt().
```

  which should end both the toolbar window and the command line window.

1.4 Corrective Maintenance

There are a couple of problems that may occur and are related to the installation of the system rather than to faulty Erlang code.

Trouble-shooting

If the Erlang system fails to start properly, the most likely cause is either that Erlang has been installed in an incorrect way, or that the advanced Erlang shell is unable to deal with the terminal type in use. The latter case can be easily identified, since Erlang in that case can be started with `erl -oldshell` (but the advanced editing features will, of course, not work). If this occurs, please submit a problem report, and be sure to report the environment $TERM definition.

A typical example of an incorrect installation is where the top level directory is incorrectly specified to the `Install` script. Check the path name specified for `Rootdir` in the installed `erl` script. If this is incorrect, rerun the `Install` script with the correct path name.

Users running the Sun OpenWindows should note that the advanced shell does not work correctly in `cmdtool` or `shelltool` if the scrollbar is activated. This is due to deficiencies in these programs. Either deactivate the scrollbar, use `erl -oldshell`, or use, for example, `xterm` instead.

Using distributed Erlang is simpler if the environment is set up to use the DNS system for host name/address lookups, but that is not required. Further details about trouble-shooting for this case can be found below.

Trouble-shooting Distributed Erlang

There are a number of things that can be wrong when starting distributed Erlang. If you cannot understand the following, ask your system administrator for help.

Distributed Erlang can be started either with the `-name` or the `-sname` flag.

To use distributed Erlang in a Wide Area Network environment it is necessary to use the `-name` flag when starting nodes. If this is done, Erlang will use the mechanism for lookup of IP-addresses of hosts as specified at the installation.

If in subsequent operations Erlang is supplied with a node name, e.g. `foobar@super.eua.ericsson.se`, Erlang will contact `epmd` (Erlang Port Mapper Daemon) at the host `super.eua.ericsson.se` in order to find the address of the node called `foobar` there.

At installation of Erlang, it is specified by the administrator installing the system only if *DNS* should be used for address lookup, or if the mechanism provided by the underlying operation system will be used (the latter may very well use *DNS* only, or a combination of *DNS* and lookups in the `/etc/hosts` file).

If *DNS* is used, all hosts involved must be properly configured for *DNS*, see the UNIX manual page `named(8)`. The following is a list of some of the things that can go wrong on a UNIX system where *DNS* is supposed to work:

1. Non existing or erroneous `/etc/resolv.conf` file. If this file is does not exist in your system, contact your system administrator, or run Erlang on a computer which has this file.

2. Host not registered with *DNS* name server. To check if your host is known to a name server use the program `/usr/etc/nslookup` (SunOS 4) or `/usr/sbin/nslookup`(Solaris 2),

```
% nslookup
Default Server: super.eua.ericsson.se
Address: 134.138.199.16
> gin
Server: super.eua.ericsson.se
Address: 134.138.199.16
Name: gin.eua.ericsson.se
Address: 134.138.199.53
> xi-term1
Server: super.eua.ericsson.se
Address: 134.138.199.16
*** super.eua.ericsson.se can't find xi-term1:
Non-existent domain
> exit
%
```

   The code above is an example session with `nslookup`. First a question about the host `gin` is asked. This is ok. Then a question about the host `xi-term1` is asked. This is a host at the site known to *NIS*, but `named` did not recognize the host. For this reason, distributed Erlang cannot run at all on the host xi-term1.

3. The portnumber of `epmd`, see `epmd(3)`, is already used by an other program. In Erlang R2A (Erlang 4.5) port 4368 is used for epmd. If this port is used by an other program, distributed Erlang cannot run. The following checks if port 4368 is already used (use `/usr/ucb/netstat` on SunOS 4, and `/usr/bin/netstat` on Solaris 2):

```
% epmd -kill
Killed
% sleep 120
% /usr/ucb/netstat -a | grep 4368
%
```

   The code above does the following:

   (a) Aborts epmd at the host. The epmd exec file is under the erlang/bin directory

   (b) Is inactive for a period to let tcp connections disappear

   (c) Checks if any other program is using the port. If the netstat command produces any output, there is an error message. This means that another program is occupying the epmd port. If possible, remove this program and try again.

   The file /etc/services must also be checked:

```
% cat /etc/services | grep 4368
% ypcat services | grep 4368
```

   If any of the above commands produce any output, there is a problem as it is not possible to choose a port number on the command line. The only solution is to try to move the other obstructing program to an other tcp/port number.

If *DNS* is not used at your site at all, Erlang should not be installed with the option of using DNS only.

The epmd program can be used for checking a host in a simlar way to nslookup.

```
% epmd -hinfo netsim-server.tei.ericsson.se
official host name: netsim-server.tei.ericsson.se
addr type = 2, addr length = 4
Internet address: 141.137.93.20
Cant't get hostbyaddr() on host 20.93.137.141.in-addr.arpa
Bad IPaddr == 141.137.93.20
% epmd -hinfo super
official host name: super.eua.ericsson.se
addr type = 2, addr length = 4
Internet address: 134.138.199.16
%
```

The above is a transcript from a session with epmd. The first host called `netsimserver.tei.ericsson.se` apparently had some problems.

If distributed Erlang is run with the `-sname` flag, it can be run in an environment where *DNS* is not running at all. If the `/etc/resolv.conf` file is not present, the resolver library routines to `gethostbyname()` and `gethostbyaddr()` will resort to reading the `/etc/hosts` file, which must then contain the IP address and names of all hosts which are to be used for distributed Erlang applications. The `/etc/hosts` file can also contain the names of non local hosts (over a WAN), but if name lookup is used over a WAN, there may be problems if the initial part of a host are the same in two different domains. If Erlang is run over a WAN, DNS is thus the recommended method.

A configuration where it is necessary to use the `-sname` flag since DNS is not needed, could be a set of target nodes that run disconnected from any network; for example, a laptop computer attached to the targets (possibly via SLIP). In that case it is not reasonable to require `DNS`.

Another example is a a set of nodes on a (not-networked) LAN with a small set of hosts, and DNS is not desirable or possible. Then the option of reading `/etc/hosts` might be appropriate.

1.5 Problem Reporting

This chapter describes how to submit a Problem Report (*PR*) on *Erlang*.

> Note:
> The procedure below is the default procedure for submitting *PR*s. If there are other means for reporting which are specific to your project or company routines and processes, those may be used.

You should *NOT* submit any PR before you have read this chapter.

Submitting Problem Reports

A *PR* is submitted with electronic mail to `support@erlang.ericsson.se`. There are three ways of submitting a problem report, by:

- manually filling in a *PR* form using your favorite editor, and mailing it to `support@erlang.ericsson.se`, or
- invoking `send-pr` from a standard shell,
- invoking `M-x send-pr'` within GNU Emacs.

Helpful Hints

- Use common sense. There is no orthodox standard for submitting effective *PR*s.
- Submit only one problem with each *PR*.
- Describe all you would want to know, if you were the person reading the report.
- Describe everything that may be different from other environments; e.g. path, *Erlang* path, *Erlang* file, but do not go too much into detail.
- Report all the facts. If you are not sure whether to include a fact or leave it out, INCLUDE IT!
- The more closely a *PR* adheres to the standard format, the quicker and more efficiently it can be handled.

The Problem Report template

The Problem Report template has the following fields,

```
>Submitter-Id:   <Your id>
>Originator:    <Your name>
>Organization:  <The name of your company>
>Confidential:  <[yes | no ] (one line)>
>Synopsis:      <synopsis of the problem (one line)>
>Severity:      < [non-critical | serious | critical ] (one line)>
>Priority:      < [ low | medium | high ] (one line)>
>Category:      <product, component or concept name>
>Class:         < [ sw-bug | doc-bug | change-request | support ] (one line)>
>Release:       <Release version, e.g. OTP R2A, Erlang 4.5>
>Environment:
        <machine, os, target, libraries (multiple lines)>
>Description:
        <precise description of the problem (multiple lines)>
>How-To-Repeat:
        <code/input/activities to reproduce the problem (multiple lines)
>Fix:
        <How to correct or work around the problem, if known
         (multiple lines)>
```

A copy of the above template is found in the root directory of your Erlang installation under the name `PR.template`. Below, each field of the template is described in detail.

Submitter-Id

The submitter field must contain your submitter identity, which you got when the system was delivered to you. If you do not have a submitter id, contact `support@erlang.ericsson.se`.

The submitter id is a unique identification code assigned by the support site.

Originator

The originator field must contain your real name.

Organization

The organization field must contain your company name.

Confidential

The confidentiality class depends on the agreement between Erlang Systems and your company. The information in this field determines whether or not the information in your PR will be made available to other Erlang-users. There are two possible values (no is the default):

Tag	*Meaning*
`yes`	The information in the PR will not be publicly available.
`no`	The information will be publicly available.

Table 1.1: Confidential

Synopsis

The synopsis should be a short summary of the problem (no more than one line). You should copy the synopsis to the Subject line of your e-mail. A good synopsis will speed up the process of routing the PR to the appropriate person.

Severity

The severity of the problem has to be decided with the aid of the guidelines below. You can choose among the following values (serious is the default):

Tag	*Meaning*
`critical`	Product is not working, essential functionality is missing, no work around is known.
`serious`	Product is not working properly, functionality is missing, work around is known
`non-critical`	Product is working, irritating behavior, mismatch in documentation

Table 1.2: Severity

Priority

The priority shows how soon you require a solution. You can choose among the following values (medium is the default):

Tag	*Meaning*
`high`	Solve as soon as possible, urgent
`medium`	Solve for next release
`low`	Solve for future release

Table 1.3: Priority

Category

The category field should contain the name of the product, component or concept where the problem was discovered.

Class

This field contains the class of the PR. You can choose among the following classes (sw-bug is the default):

Tag	*Meaning*
`sw-bug`	Software bug
`doc-bug`	Documentation bug
`change-request`	Change request
`support`	Support request
`duplicate`	Duplication of a previous PR
`mistaken`	Statement that previous PR was incorrect

Table 1.4: Class

Release

The release field should contain the release version number of the product, component or concept on which you are reporting a problem.

Environment

This field must contain the description of the environment where the problem occurred; such as machine architecture, operating system, host and target.

Description

Describe the problem in as exact terms as possible in the field between the lines `Description:` and `How-To-Repeat:`. For information on what to include, see *Helpful Hints*.

How-To-Repeat

If you know how to reproduce the problem, describe how it is to be done between the lines `How-To-Repeat:` and `Fix:`.

Fix

If you know how to resolve the problem, describe how it is to be done below the line >`Fix:`.

Checking the fields

Before submitting the PR, please check all fields to see if they are correct.

If you have invoked `send-pr` from a shell, or from within GNU Emacs, check that the information automatically filled-in by `send-pr` is correct.

Submitting a PR

A PR is submitted by mailing the PR to `support@erlang.ericsson.se`.

Submitting a Problem Report from a Shell

You can use `send-pr` from a shell (sh, csh, tcsh, etc.) in order to fill in and submit a PR, provided you have installed a `send-pr` package.

When `send-pr` is invoked from a shell, it will try to determine what editor to use. The values for the environment variable `EDITOR` or `VISUAL` are looked up and are used for the choice of editor. If neither is found, `vi` will be used as the editor for the PR.

If `send-pr` is used from a shell, the editor is only provided with a template, and all the editing has to be done by the user. When you have finished the editing, save the file and quit the editor.

The PR is then submitted by pressing `s` when a prompt is shown.

Submitting a Problem Report from GNU Emacs

It is possible to fill in and submit a PR from within GNU Emacs. See the GNU info pages on GNATS for more information.

List of Tables