System Architecture Introduction

version 4.8

OTP Team

1997-05-02

Contents

Chapter 1

System Architecture Introduction

1.1   Introduction

Erlang is a programming language specifically designed to build fault-tolerant, distributed systems which can contain a large number of concurrent processes. The Erlang Development Environment described in this User's Guide contains the following building blocks:

- the Erlang runtime system
- an integrated, window-based interface for program development
- application development tools.

Erlang is a high-level functional language. The language combines important attributes of declarative languages with constructs for supporting concurrency, distribution, and error-detection. Erlang programs are made up of functions which are grouped into modules. Functions spawn processes which are the executing elements of an Erlang system. Processes communicate by sending and receiving message via ports, which themselves behave like processes. A built-in distribution mechanism enables designers to create a system whose processes may run on different computers. Erlang handles fault detection and recovery in distributed systems and has fault recovery schemes, which can be customized. These facilities are described in Chapter 3; Design Principles.

The Development Environment includes several facilities for creating interfaces between applications written in Erlang and other programming languages. These include:

- an open_port mechanism
- a socket library
- a C/C++ interface generator
- an erl_interface library which includes functions to create C structures which behave like an Erlang node.

The Development Environment contains numerous tools to support the tasks of developing and testing applications. These tools include:

- editor
- compiler
- debugger
- C Interface generator
- application monitor
- process manager
- coverage analyzer
- profiler
- graphics interface
- ASN.1 compiler
- cross-reference tool
- parser generator
- measurement handler.

These tools are described in detail in Chapters 2, 6 and 7 of this User's Guide. Refer also to the Reference Manual.

The Erlang Runtime System

The Erlang runtime system is made up of the following parts:

- the Erlang virtual machine
- the kernel
- the standard library.

The Erlang Virtual Machine

The Erlang virtual machine runs on top of a host operating system. The Erlang runtime system frequently runs as a single process in the host operating system. However, in many of the operating systems that support Erlang, it is possible to run several completely unrelated Erlang virtual machines in parallel. The following support is provided for Erlang programs:

- a consistent operating system interface on all platforms
- memory allocation and real-time garbage collection
- light-weight concurrency and support for thousands of simultaneous tasks
- transparent co-operation between all computers in the system
- location and encapsulation of run-time errors
- supervision of run-time code as it loads, when it is replaced, and while it is linked.

The Kernel

The `kernel` is always the first application to be started. It provides low-level services which are necessary for an Erlang system to start, to participate in a distributed system, to handle errors, and to perform IO operations.

Refer to the Reference Manual, section `kernel` where each module is fully described. Chapter 4 of this User Guide also has information about these services.

Standard Library

The standard library contains a large number of re-usable software modules which greatly aid the Erlang system developer. Many of these modules are specially adapted to programming concurrent, distributed systems.

The Reference Manual describes these modules in detail. Modules which solve common programming tasks are also described in Chapter 3 of this User's Guide. These include `gen_server`, `gen_event`, and `gen_fsm`

Scope and Purpose

This manual describes the Erlang Development Environment. The major focus is twofold:

- to describe the development tools which are available to the user
- to describe the design principles for developing Erlang systems.

All of the chapters provide numerous examples which illustrate the concepts and the theories described and a glossary defines the terminology used.

Exclusions

This User Guide assumes that the reader is familiar with the Erlang programming language and does not explain how to program in Erlang. References to programming manuals are listed at the end of this chapter.

About This Book

The chapters in this User Guide are independent of each other and can be read in any order.

- Chapter 2: "Getting Started with Erlang" describes the development environment and introduces the reader to tools such as the compiler and debugger.
- Chapter 3: "Design Principles" describes how to develop Erlang systems, including distributed systems. The standard behaviours are described and numerous examples illustrate how to apply these behaviours to typical applications. This chapter also describes how to structure, create, configure, and test an application.
- Chapter 4: "System Principles" describes the Erlang system in some detail. It contains information about starting and stopping Erlang, accessing command line arguments, creating a boot file, code loading strategies, and Erlang applications.
- Chapter 5: "Operation and Management Principles" describes the model for operation and maintenance of sub-systems.
- Chapter 6: "Interface Libraries" describes how to utilize special features of the Erlang development environment, such as the erl_interface library, the C/C++ Interface Generator, the Java Interface, and the graphics system.
- Chapter 7: "Tools" describe to special tools included with the Erlang distribution. These include the Toolbar, Erlang for Emacs, the Process Manager, the Coverage Analyzer, the Profiler, the Application Monitor, and the Cross-Reference Tool.

Typographical Conventions

The following typographical conventions are used in this user's guide.

convention	where used
command	To show menu selections and equivalent command line entries To show keyboard entries at system prompts
`code`	To highlight Erlang code, module and function names, arguments, variables, and file names.

Table 1.1: Examples of Typographical Conventions

Where to Find More Information

Refer to the following documentation for more information:

- the "Erlang Extensions Since 4.4" User's Guide

- the Reference Manual
- the "Erlang Embedded Systems" Book
- the "Database Management" Book
- the "System Architecture Support Libraries (SASL)" Book
- the "Object Request Broker and IDL Compiling" Book
- the "Event and Alarm Handling (EVA)" Book
- the "SNMP" Book
- the "Installation Guide"
- the "Embedded Systems" Book
- the "Erlang Extensions Since 4.4" Book
- the "Abstract Syntax Notation,ASN.1" Book
- Concurrent Programming in Erlang, 2nd Edition, ISBN 0-13-508301-X.

List of Tables

Chapter 1: System Architecture Introduction