Introduction

version 5.6

Typeset in LaTeX from SGML source using the DocBuilder-0.9.8.4 Document System.

Contents

Chapter 1

Introduction

1.1 Introduction

1.1.1 Erlang and OTP

Erlang is a general-purpose programming language with built-in support for concurrency, distribution and fault tolerance.

OTP (Open Telecom Platform) is aimed at providing time-saving and flexible development for robust, adaptable telecom systems. It consists of an Erlang runtime system, a number of ready-to-use components mainly written in Erlang, and a set of design principles for Erlang programs. Since Erlang and OTP are closely interconnected the term Erlang/OTP is normally used instead of OTP.

1.1.2 Erlang/OTP

Erlang Runtime System

The Erlang runtime system (ERTS) is made up of an emulator running on top of the host operating system, a kernel providing low-level services such as distribution and I/O handling, and a standard library containing a large number of re-usable modules.

The OTP design principles provides the user with a way to structure the system based on a concept called application. An OTP application is a way to package a system component and is either a set of library modules or a supervision tree. A supervision tree is a hierarchical tree of processes used to program fault-tolerant systems. The processes are easiest implemented using behavior modules which are formalizations of design patterns. The standard library includes behavior modules for supervisors, servers, state machines and generic event handlers. In chapter 4 “OTP Design Principles” the design principles are explained in detail.

OTP Components

The OTP components can be divided into six categories:

- Basic Applications - Basic Erlang/OTP functionality.
  - *Compiler* A compiler for Erlang modules.
  - *Kernel* Functionality necessary to run Erlang/OTP itself.
  - *SASL* (System Architecture Support Libraries) A set of tools for code replacement and alarm handling etc.
  - *Stdlib* The standard library.
- Operations and Maintenance - OAM both of the system developed by the user and of Erlang/OTP itself.
  - *EVA* A multi-featured event and alarm handler.
  - *OS_Mon* A monitor which allows inspection of the underlying operating system.
  - *SNMP* SNMP support including a MIB compiler and tools for creating SNMP agents.
- Interface and Communication - Interoperability and protocols support.
  - *Asn1* Support for ASN.1.
  - *Comet* A library that enables Erlang/OTP to call COM objects on windows
  - *Crypto* Cryptographical support
  - *Erl_Interface* Low level interface to C.
  - *GS* A graphics system used to write platform independent user interfaces.
  - *Inets* A set of services such as a web server and a FTP client.
  - *Jinterface* Low level interface to Java.
  - *SSL* Secure Socket Layer (SSL),interface to UNIX BSD sockets
- Database Management.
  - *QLC* Query language support for Mnesia DBMS.
  - *Mnesia* A heavy duty real-time distributed database.
  - *ODBC* ODBC database interface.
- CORBA services and IDL compiler.
  - *cosEvent* Orber OMG Event Service.
  - *cosNotification* Orber OMG Notification Service.
  - *cosTime* Orber OMG Timer and TimerEvent Services.
  - *cosTransactions* Orber OMG Transaction Service.
  - *IC* IDL compiler
  - *Orber* A CORBA object request broker.
- Tools.
  - *Appmon* A utility used to view OTP applications.
  - *Debugger* For debugging and testing of Erlang programs.
  - *Parsetools* A set of parsing and lexical analysis tools.
  - *Pman* A process manager used to inspect the state of an Erlang/OTP system.
  - *Runtime_Tools* Tools to include in a production system.
  - *Toolbar* A tool bar simplifying access to the Erlang/OTP tools.
  - *Tools* A set of programming tools including a coverage analyzer etc.
  - *TV* An ETS and Mnesia graphical table visualizer.

1.1.3 Scope and Purpose

This documentation describes the Erlang runtime system, the OTP applications and the OTP design principles. It assumes that the reader is familiar with the Erlang programming language and does not explain how to program in Erlang. The language is described in *Concurrent Programming in Erlang, 2nd Edition*, ISBN 0-13-508301-X.

1.1.4 About the Erlang/OTP Documentation

Structure of this Book

The documentation is divided into eight parts. This book, *Erlang 5.1/OTP R8 System Documentation, EN/LZT 108 4095 R2*, is the starting point of the documentation and contains information about the Erlang programming language and runtime system, the OTP design principles, and how to install and configure Erlang/OTP.

- Chapter 2: “Getting Started with Erlang” describes the Erlang runtime system and introduces the reader to tools such as the compiler and debugger.
- Chapter 3: “Erlang Extensions Since 4.4” lists all extensions added to the Erlang programming languages since the latest version of the book *Concurrent Programming in ERLANG*.
- Chapter 4: “OTP Design Principles” describes a way to structure Erlang code in terms of applications and supervision trees. The standard behaviors are described and examples illustrate how to apply these behaviors to typical applications.
- Chapter 5: “Installation Guide”gives guidelines on how to install Erlang/OTP on UNIX or Windows.
- Chapter 6: “System Principles” describes the strategies and options, which are available to start an Erlang/OTP system. This chapter also provides a brief description of the applications included in an Erla ng/OTP system.
- Chapter 7: “Embedded Systems” is a supplement to “Installation Guide”. t describes issues that are specific for running Erlang/OTP on an embedded system.
- Chapter 8: “Operation and Management Principles” describes the model for operation and maintenance of sub-systems.
- Chapter 9: “Tutorial” gives an orientation of the different interoperability mechanism, which can be used when integrating an Erlang program with a program written in an other programming language.

Typographical Conventions

The following typographical conventions are used in the documentation.

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.

List of Tables