ERLANG Specification*(Assignment 3)*

Abhijit Mohanta
MS-CSE
Roll no: 201681001

1 Introduction

Erlang is used as a general purpose programming language. It has been developed to support concurrency, distribution and fault tolerance.

A. Erlang is mainly suitable for distributed, reliable, soft real-time applications (e.g. video streaming).

- Internet applications servers, e.g. a mail transfer agent, HTTP server, etc.
- Database applications where soft real time behavior is required. RIAK and CouchDB are using Erlang for database purpose.
- In telecommunication systems, e.g. switch controlling.
- Telecommunication applications.
- Web-based dashboards that deal with real-time data.
- Service oriented software architecture, e.g., RabbitMQ is using Erlang.Erlang is meant to solve these types of problems.

B. Here are the few common situations where Erlang is not a suitable language to choice.

- Erlang is not suitable for signal processing, image processing, sorting large volumes of data.
- Another problem is a wide interface to existing C code, e.g., implementing operating system device drivers.
- Erlang is not suitable developing logic for complex protocol.
- Web applications that do not include real-time communications.
- Erlang is not good for string operations like transformations, parsers, etc.
- It is not good for desktop GUI applications.

C. ERLANG design issues that are not good. Here are the few issues with Erlang designing.

- In Erlang records suck and there are no structure or map data structure available in Erlang.
- Erlang sucks at managing memory.
- Erlang is not general purpose language. It hates state specially shared state.To deal with shared state in Erlang provides feature called "Erlang Term Storage" and also provides a Judy array.
- Erlang syntax is atrocious.
- Immutable state is not necessary for Erlang-Style Concurrency.
- Single assignment is just as problematic as destructive assignment.
- The standard library is inconsistent.

2 Data Types

Erlang provides a number of data types, like Terms, Number, Atom, Bit Strings, Reference, Fun, Port Identifier, Pid, Tuple, Map, etc.

- **Terms:** Term is a piece of data of any data types.
- **Number:** integers and float are two types of data available in Erlang. Also, there are two Erlang- specific notation, i.e., $char and base#value.
- **Atom:** An Atom is a constant with a name.It is enclosed in single quotes only if it begin with an upper-case letter or if it contains other characters than alphanumeric characters.
- **Fun:** It creates an anonymous function and pass the function itself.
- **Port Identifier:** It identifies the Erlang port. open port / 2 is used to create port.
- **Reference:** It's unique in an Erlang runtime system, which is created by calling make_ref/0.
- **Fun:** It creates an anonymous function and pass the function itself.
- **Pid:** pid means process identifier that identifies a process.
- **Touple:** It is a compound data type which is having a fixed number of terms.
- **Map:** It's a compound data type with a variable number of key-value assigned. Each value assigned in map is called association pair.
- **List:** It's a compound data type with a variable number of terms.
- **String:** It is enclosed with double quotes, but it's not a data type in Erlang language. Instead, a string is considered as a list in Erlang.
- **Record:** It is used to sort affixed number of elements and it is a data structure. It is same as the struct in C.

3 Pattern Matching

Pattern matching occurs during function call, case- receive- try- expressions and match operator expressions. In pattern matching variables are bound to values. Left-hand side pattern is matched against a right-hand side and if match is found, any unbound variable in the pattern become bound, otherwise run-time error occurs.

4 Module

Module contains a sequence of a attribute's sequence and function declarations. Module terminated by a period.

Module Attributes: A module attribute defines the properties of a module.

5 Comments

It can be placed anywhere in a module except within strings and quoted atoms. A comment begins with the character %.

6 Function

A function declaration is a sequence of function clauses which is separated by semicolons, and terminated by full stop or period. Function clause consist of a clause head and clause body. Clause head consist of a function name, list of arguments, and an optional guard sequence which begins with the keyword when. The function name should be an atom. Also, the body of a clause contains a sequence of expressions and they are separated by comma.

Tail Recursion: If the last expression of a function body is function call, then that is called tell recursion.

7 Compilation and Code Loading

The compilation process and loading is system-dependent but it is not a language issue.

Compilation: Erlang program is compiled to object code and the compiler can generate a new file which consist of object code. The compiler is situated in the compile module. Erlang shell supports the command c(Module) which compiles and loads Module. Also, there is another module called make, which is similar to UNIX type Make function.

8 Distributed Erlang

Distributed Erlang system contains a number of Erlang runtime systems which communicates with each other and such runtime system is called a node. Message passing between processes at various nodes, also links and monitors, are transparent in case of usedpid. Registered names are local to each node.The distribution process is implemented with the help of TCP/IP sockets.

References

[1] http://erlang.org/doc/index.html

[2] https://www.tutorialspoint.com/erlang/index.htm

[3] http://learnyousomeerlang.com

[4] http://www.ibm.com/developerworks/library/os-erlang1/