

Symbols

-- operator 373
_ (don't-care pattern) 51
_app suffix 124
_sup suffix 126
; 54
! 12, 77–78, 106
. 42, 54
\$ 29, 35, 51
\$'_ 229
\$' 166
\$\$' 229
\$1, \$2, and so on 229
\$ROOT 250
[] 33
@ 32, 334
@ tags in EDoc comments 102, 109
/ 30
/= 37
\ 63
29, 70
% 42–43
+ 41
++ 34, 52, 89–90, 373
+W option 174
< 36
<- 66, 69
<<...>> 31
<= 37, 69
= 47, 49
:= 37
/= 37
== 36–37
> 42

> 36
>= 37
| 33
|| 66
~n 53
~p 53, 174
~w 53

Numerics

100 Continue 273, 285
200 OK 286–287, 289
404 Not Found 289
501 Not Implemented 289

A

accumulated time 365
accumulator parameter 85, 88, 90
ACID 226
active application 120
vs. library application 121
{active, false} 280
{active, once} 281, 284
Ada 29
after 64, 77
alias pattern 210
Amnesia 220
Apache Commons Logging 354
Apache log4j 354
Apache ZooKeeper 354
API 104, 126
.app file 121–122, 129, 243, 248
loading 245
parameters 123

append operator 34
application 120–125
.app file, loading 245
active. *See* active application
behaviour 123–125, 152, 244
implementation 154
controller 144, 244–245
dependencies 123, 244, 246, 249
indirect 244
directory layout 121, 153
from system viewpoint 243–245
introduction 119
library. *See* library application
master processes 134, 245
meaning of, in OTP 120
metadata 120, 122–123, 130, 154, 243
root supervisor 120
skeleton, creating 153–157
start types 245
starting 129
structure 243
organization of 120–122
visualizing 132
application:get_env/2 235, 287
application:start/1 129, 245
application-level API 167
applications parameter 123
apply/3 116, 269
using Mod:Func(...)
instead 375

Appmon 132–137, 252
 GUI 133–136
 in Toolbar 146
 in WebTool 136
 killing a process in 136
 menus 133
 process information
 window 135
 tracing in 136
 appmon:start() 133
 arithmetic 30
 arity of a function 40
 Armstrong, Joe 92, 358
 array 372
 array module 41
 arrows 37
 ASCII 29, 31
 ASN.1 122
 assertion 234
 assignment 47
 vs. pattern matching 49
 associative array 207
 asynchronous cache. *See* distributed cache, asynchronous
 asynchronous
 communication 214–215
 broadcasting 217
 message passing 191
 atom 370–371
 atom table 370
 efficiency of 31
 introduction 31
 limit on atom table size 32
 maximum length of 32
 not garbage collected 371
 size 369
 atom_to_list(A) 35
 {atomic, Data} 229
 atomicity 227
 definition 8
 auth:get_cookie() 198
 auto-imported functions 41

B

badarg 62
 badmatch 49, 118, 234
 bag 81, 224, 228
 duplicate 81
 band 30
 bang operator 12
 base case 83, 86, 88
 minimizing 88
 baseline, for performance
 tuning 359

batch job 24
 BEAM 17–20
 tag bits 369
 .beam 43–45
 .beam file
 debugging information 141
 beautiful code 358
 behaviour
 API 103–109
 attribute 102
 callbacks 102, 109–116, 276
 components of 97–98, 274
 container 98, 138, 244, 274
 gen_event 171, 179, 181, 188
 creating custom event
 stream with 183
 example 180
 gen_server 158
 implementation 97
 instantiating 99
 interaction with compiler 275
 interface 97
 introduction 97–100
 message wrapper 107
 module header 101–103
 module layout 100
 supervisor 123, 126, 155
 behaviour declaration 275
 behaviour_info/1 275
 BIF. *See* built-in function
 big, specifier in bitstring 68
 big-endian 302
 bignum 370
 boxed representation 370
 introduction 29
 size 369
 Bigtable 333
 bin_opt_info compiler flag 372
 binaries, pattern matching
 on 289
 binary 30
 heap binary 371
 large 371
 reference-counted binary 371
 representation of 371
 size 369
 specifier in bitstring 68
 binary_to_term/1 299
 binding variables 47, 49
 bit syntax 67–68
 bit_size/1 374
 bits, specifier in bitstring 68
 bitstring 31
 building 67–68
 comprehension 69

length 67
 representation of 371
 size 369
 black box 246
 blocking communication 215
 bnot 30
 body recursion, efficiency
 of 375
 Bogdan 17
 Boolean function 54
 Boolean switch 57
 -boot 251
 .boot file 249, 255
 regenerating 250
 boot script 249–250
 booting 249, 252
 bor 30
 bottleneck 346
 identifying with profiling 360
 synchronization
 bottleneck 361
 box with a name 47
 boxed representation 370
 brain damage 82
 BREAK menu 27
 breakpoint 141
 conditional 143
 broadcasting, and asynchronous
 communication 217
 brutal_kill 128, 156
 bsl 30
 bsr 30
 built-in function
 can't be call-counted with
 cprof 363
 performance caveats
 373–374
 bulwark 14
 bxor 30
 byte 30
 byte_size/1 374
 bytes, specifier in bitstring 68

C

C
 array vs. ETS table 80
 C node 292
 c_src directory 344
 dynamically allocated
 memory 314
 external (global)
 variables 313
 external port program
 300–312

- C (*continued*)
 - implementation of ETS
 - tables 80
 - implementing built-in functions in 41
 - looping 82
 - preprocessor 72
 - shared library 293
 - syntax 21
 - unsafe code 295
- c_src directory 300, 344
- c(...) 43, 73
- C10K problem 19
- cache
 - architecture and process scheduling 18
 - asynchronous. *See* distributed cache, asynchronous
 - creating 157–169
 - design 151–153
 - distributing 214–219
 - in HBase 341
 - synchronous. *See* distributed cache, synchronous
 - weak consistency 217
- calendar module 161
- call 111
- CamelCase 46
- case expression 56–57
- cast 112
- cast/2 185
- catch 63, 65
- cd(Dir) 26
- Cesarini, Francesco 92
- character code 29, 31
 - writing with string 34
- character, printable 35
- child process 12
- child specification 128
- .class file 354
 - priv directory 344
- .class files 337
- clause 54
 - guards 55
 - in case expression 57
 - in if expression 58
 - mutually exclusive 56, 89, 375
 - order 83
 - order of 92
 - ordering of 89
 - pattern-matching
 - compilation 375
 - selecting with pattern matching 56
 - selection 54
 - selection and efficiency 375–376
- Clojure 9
- closure 60–61
- cloud environment, node
 - discovery 197
- cluster 193
 - and asynchronous communication 217
 - communication overhead 195
 - connecting to 234
 - creating 195–197
 - fully connected 193
 - number of nodes 195
- clustering 193–203
- cmd.exe 45
- code
 - beautiful 358
 - instrumenting 187
 - loading at boot time 252
 - procedural 82
- code module 44
- code path 129, 249
 - introduction 44
- code:get_path 44
- code:lib_dir/1 130
- code:priv_dir() 122
- code:priv_dir/1 298
- code:root_dir() 254, 312
- collection, iteration over 90
- com.ericsson.otp.erlang 337
- command-line Erlang
 - compiler 44
- comment 42–43
 - file-level 101
 - with EDoc annotations 101
- Common Test 117
- communication
 - asynchronous. *See* asynchronous communication
 - blocking 215
 - bottleneck 346
 - model 191–192
 - strategies 214–216
 - synchronous. *See* synchronous communication
 - transparent 191
- comparison operators 36–37
- compiler
 - bin_opt_info flag 372
 - erlc 44
- compiling
 - conditional. *See* conditional compilation
 - module, from the shell 43
- compound data structure,
 - constructing 33
- comprehension 67
 - See also* bitstring comprehension
- concurrency 5–13
 - cheapness of 11
 - profiling 361
- Concurrent Programming in Erlang* 92
- Concurrent Prolog, futures 10
- conditional compilation 74
- config 251
- .config file 251
- configuration file 204, 250–251
- configuration settings 236
- Cons 129
- cons 90
- cons cell 38
- consistency 227
- console window 201
- constant space 86
- consumer, in resource
 - discovery 204
- contact node 234
- container 99, 274
- Content-Length 273, 285
- Content-Type 272
- control flow 49
- controlled shutdown of Erlang
 - system 26
- cookie file 198
- cookie, setting when creating a
 - node 334
- copying 16
 - and processes 7
 - overhead of 10
- CouchDB 193
- cover (profiling tool) 361
- cover module 363
- covert channel 382
- cp (class path) flag 337
- cprof 361–363
- CPU
 - consumption 359
 - multiple 5
 - time 133
 - measuring usage 138
 - topology and process scheduling 18
- crash report 178
- CRUD 164, 286
- Ctrl-Break 27, 202
- Ctrl-C 27, 202
- Ctrl-G 27, 201–202

Ctrl-N 24
 Ctrl-P 24
 cubic time 90
 curl 271
 current directory 44
 current function 138
 currently running jobs 202
 Cygwin, and HBase 341

D

-D 74
 daemon
 EPMD 197
 running as 24
 data model 220
 data structure
 decomposing 49
 mapping into Java 335–336
 traversing 86
 data types 29–39
 abstract 33
 built in 29
 primitive 369–373
 size of 369
 test 55
 data, checking untrusted
 input 115
 database
 distributed 219
 schema 222
 dataflow variables 10
 dbg 136, 363
 DEBUG 74
 debug_info compiler flag 141
 debuggability, and referential
 transparency 382
 debugger
 attached process window 141
 source-level 140–143
 Debugger, in Toolbar 146
 debugger:start() 140
 debugging
 information in .beam file 141
 via BREAK menu 27
 declaration, introduction 42
 declarative programming, power
 of 349
 deep list 115
 DEFAULT_LEASE_TIME 158
 define 72
 delegate 59
 DELETE request 273
 delete/1 162, 268
 delivery guarantee 78

dependencies, eliminating 6
 dependency 244
 description parameter 123
 -detached 252
 development and referential
 transparency 382
 Dialyzer 37, 323, 374
 and case expressions 57
 dict module 41, 207, 209
 dict:find/2 209
 dictionary (data structure) 207
 dirty operation, in Mnesia 227
 dirty_read 228
 disc_copies 225
 disk_only_copies 225
 distributed cache
 asynchronous 216–219
 in Mnesia 230
 making aware of other
 nodes 233–236
 synchronous 216–219
 latency 218
 weak consistency 217
 distributed database 219
 distributed programming 190
 distributed system,
 nondeterminism 192
 distributed table 219
 distributed transaction 218
 distribution 7, 16–17
 fundamentals 190–193
 net_adm:ping/1 196
 transparent 8
 div 30
 .dll 320
 DNS, and distributed mode 194
 do_sum 85
 doc directory 121
 documentation
 generating with EDoc 130
 documentation, generating with
 EDoc 130
 don't-care pattern 51
 dot notation, for accessing
 record fields 70
 double precision 30
 double quotes 34, 51
 driver_alloc() 314, 317
 driver_entry 316
 driver_free() 314, 317
 DRIVER_INIT 316
 driver_output 319
 durability 227
 dynamic-link library (DLL) 320

E

E programming language,
 futures 10
 ebin 45, 129
 directory 121–122, 129
 EC2, node discovery 197
 editor, comments in 43
 EDoc 101, 130
 @ tags 102
 comments, type specification
 in 109
 eggs, breaking 5
 ei 292
 and plain ports 295
 decoding and encoding
 Erlang terms 304–307
 header file 301
 ei_decode_version 306
 ei_encode_tuple_header 311
 ei_get_type 306
 ei_x_buff 302, 305
 index position 311
 ei_x_encode_atom 306
 ei_x_encode_empty_list 311
 ei_x_encode_tuple_header 306
 ei_x_new_with_version 305
 Emacs
 comments in 43
 key bindings 24
 embedded mode 252
 encapsulation 166
 end 57, 59, 63
 enif_alloc 324
 enif_inspect_binary 326
 enif_make_list_from_array 325
 enif_make_tuple_from_array
 325
 enum 31
 eper 360
 EPMD 197, 339
 daemons 197
 eprof 363
 equality comparison 37
 exact equality 37, 48
 exact inequality 37
 Ericsson 92
 .erl 42–44
 erl 24, 45, 380
 ERL_BINARY_EXT 306
 ERL_COMPILER_OPTIONS
 environment variable 372
 erl_driver API 315
 callback functions 317
 erl_driver.h 315

- Erl_Interface 292, 300
- erl_nif 292
- erl_nif API 322, 324
- ERL_NIF_INIT 326
- ERL_NIF_TERM 326
- erl_nif.h 323
- erl_parse 269
- erl_scan 269
- erl_tar 253
- erl_tar:create/3 256
- erl_tar:extract/2 257
- erl.src 255
- Erlang
 - books 92
 - code, straightforward 368
 - communication model
 - 191–192
 - compiling from source
 - code 379
 - configuration problems,
 - resolving 380
 - data structure, mapping into
 - Java 335–336
 - data types, sizes of 369
 - decomposing data
 - structures 349
 - distributed message
 - passing 199–201
 - distributed mode 194
 - distribution 197–199
 - distribution protocol 339
 - external term format 305
 - icon (in Windows) 195
 - installing
 - on Linux, Unix, or Unix-like system 379–380
 - on Mac OS X 379–380
 - on Windows 379
 - via package manager 379
 - integrating with Java using
 - Jinterface 334–340
 - language caveats 368–378
 - magic cookies 198
 - traffic, tunneling 198
 - version of running system 27
 - VM 193
 - as a node 194
 - website 92
- Erlang icon 24, 45
- erlang module 41
- Erlang Port Mapper Daemon.
 - See* EPMD
- Erlang Programming* 92
- Erlang Runtime System. *See* ERTS
- Erlang Term Storage 79, 163
- Erlang VM 17
- erlang:'+'/2 41
- erlang:get_stacktrace() 64
- erlang:hibernate/3 378
- erlang:load_nif/2 322
- erlang:nif_error/1 323
- erlang:port_connect/2 294
- erlang:raise/3 64
- erlang:self() 41
- erlang:trace/3 136
- .erlang.cookie 198
- Erlang/OTP 3
 - installer 255
- Erlang-HBase bridge 351–355
- erlang-questions@erlang.org 92
- erlc 44–45, 73–74, 129, 380
 - providing path to behaviour
 - .beam file 290
- ErlDrvData 319
- ErlDrvEntry 316
- erlIDE, comments in 43
- ErlNifEnv 324, 326
- Erlware 149–150
 - application skeleton,
 - creating 153–157
 - cache design 151–153
 - caching 150
 - web server 150
 - working cache 157–169
- error handling 7
- error logger 62
- error report 176
- error_logger 172, 180
 - API 173–175
 - events 181–183
 - report type 174
- error_logger:error_msg/1 173
- error_logger:error_report/1
 - 174
- error_logger:info_msg/1 173
- error_logger:info_report/1 174
- error_logger:warning_msg/1
 - 173
- error_logger:warning_report/1
 - 174
- error_msg 174
- error_report 182
- ERTS 17, 20, 163
 - including in release 253
 - version 247–248
- erts 253
 - atom 248
- etop 360
- ETS
 - match pattern 167
 - match_delete function 164
 - matching functions 166
 - named table 164
 - pattern match syntax 166
 - switching from, to
 - Mnesia 230–233
 - table handle 164
- ets 80
- ETS table 79–81, 163
 - and Mnesia 224
 - creating 80
 - design philosophy 79
 - lookup 165
 - set 165
 - using patterns to search
 - tables 166
 - viewing in TV 144
- ets:insert/2 165
- ets:lookup/2 81, 165
- ets:match_delete/2 166
- ets:match/2 167
- ets:new/2 80, 164
- EUnit 117
- event handler 180, 183
 - custom 179–183
- event handling 170
 - gen_event 179–180
- event manager 179, 184
- event stream
 - API 184–185
 - custom 183–189
 - subscribing to 188–189
 - integrating into
 - application 185–187
 - protocol 185
- event, posting 187
- exception 61–65
 - catching everything 63
 - classes 62
 - cleaning up side effects 64
 - handling 13, 62
 - rethrowing 64
 - stack trace 64
 - throwing (raising) 62
- execution time, profiling with
 - fprof 363–368
- ExecutorService 348
- 'EXIT' 14
- exit 62
- exit signal 13, 78
 - trapping 14–15, 76
- exit(normal) 62
- exit/1 62, 76

exit/2 76
 Expect header 273
 100-continue 285
 exponential time 90
 export declaration 42
 expression, entering 24–25
 external term format 305

F

f() 47
 f(X) 48
 factory 156
 failing early 55
 false 32
 and Boolean function 54
 in case expressions 57
 fault isolation 280
 fault tolerance 13–16, 191
 adding with supervisors
 125–129
 fetch_resources/1 209
 fetch/1 161
 FILE 73
 file:consult/1 364
 file.hrl 73
 filtering with list
 comprehension 66
 fire and forget 214
 firewall problems 196
 float 30
 boxed representation 370
 precision 370
 single precision, lack of 30
 size 369
 specifier in bitstring 68
 flow control, and TCP
 sockets 279
 foreign function interface
 (FFI) 292
 foreign key 163
 fprof 361, 363–368
 vs. cprof 361
 fully connected network 193
 fully qualified domain
 name 194
 fun 36, 58–61
 anonymous 59–61
 dependency on module
 version 60
 format 36
 local alias 58
 remote alias 59
 size 369

function 52–56, 96
 anonymous functions 59–61
 arity of 40
 as data 36
 auto-imported 41
 body 42
 Boolean 54
 built-in 41
 call 374–376
 current 136
 definition 42
 function clauses 54–55
 head 42
 higher order 59
 introduction 39–46
 recursive 82
 non-tail 84
 tail 84
 tips for writing 86
 side effects 53
 function calls
 counting with cprof 361–363
 dynamic 269
 function_clause 54
 functional programming 20
 futures 8, 10

G

garbage collection 19, 56
 avoiding by setting heap
 size 377
 definition 366
 pause times 20
 GC 19
 gcc 300, 312
 gen_event 125, 171
 creating custom event stream
 with 183
 custom event handler
 179–183
 handlers 179
 implementing 179
 instance 179, 184
 interface 179
 start_link 180
 gen_event:add_handler/3 180
 gen_event:notify/2 185
 gen_event:start_link/1 184
 gen_fsm 125
 gen_server 97, 125, 157, 175,
 207, 348
 callbacks 160
 calling itself 117
 library functions 104

main loop 138
 managed by simple-one-for-
 one supervisor 262
 server timeout 267
 start_link 267
 stopping 112
 timeout between requests 113
 gen_server:call/2 104, 107–108,
 110, 218
 gen_server:call/3 107, 214
 deadlock 210
 gen_server:cast/2 104, 108, 110
 between gen_server
 processes 210
 gen_server:handle_info/2 269
 gen_server:loop/6 138
 gen_server:start_link/3 160
 gen_server:start_link/4 104,
 108, 110
 gen_tcp 111
 gen_tcp:accept/1 267
 gen_tcp:read() 280
 gen_web_server 274, 290
 container calling implementa-
 tion module 285
 implementation module 287
 parallel instances of 283
 source code 276
 structure of 274
 top level supervisor 279
 generator 66
 generic server 97
 GET request 271
 get_count() 105
 get_count/0 107
 get/1 79
 GHC 9
 Gleader 182
 Google 20
 and slow-serving pages 150
 Bigtable 333
 GOTO 8
 graphical tools 132
 Gregorian seconds 161
 guard 55
 in case expressions 57
 gws_connection_sup 275, 278
 gws_server 275, 278, 280–281
 .gz 253
 gzip 253

H

h() 26
 Hadoop 333

- Hadoop Common 340, 354
 - handle_call/3 110–111, 209
 - handle_cast/2 110, 112, 210
 - handle_data/3 268
 - handle_event/2 179, 188
 - handle_info 285
 - handle_info/2 110, 113, 175
 - hash table 163, 224
 - in ETS 165
 - Haskell 9
 - futures 10
 - HBase 333
 - cache 341
 - configuring 341
 - Erlang-HBase bridge 342–355
 - Erlang code 342–344
 - HBaseConnector class 344–346
 - HBaseNode class 346–349
 - HBaseTask class 349–351
 - running 353–355
 - installing 340
 - integrating with Erlang 333
 - interface with Simple Cache 336
 - Java API 344
 - Java byte arrays 345
 - map 341
 - NavigableMap 345
 - shell 341
 - SSH requirement 341
 - table, creating 341, 345
 - hbase_server 343
 - HBaseConfiguration 345
 - HBaseConnection 351
 - HBaseConnector 342, 344–347
 - HBaseNode 342, 346–349
 - HBaseTask 342, 348–351
 - head, of a list 38
 - header file 73, 121
 - and record declarations 71
 - macro definitions in 73
 - header word 372
 - heap binary 371
 - heap size, initial, setting 377
 - heart 245
 - heart process 245
 - help() 26
 - hi_server 287–288
 - implementing RESTful interface 287
 - highly available systems 19
 - home directory 198
 - Homebrew 379
 - HOMEDRIVE 198
 - HOMEPATH 198
 - host machine, nodes running
 - on 194
 - .hrl file 73, 121
 - HTable 345
 - HTTP 271–274
 - 100 Continue 273
 - and REST 286
 - body 271, 285
 - DELETE request 273
 - empty line (end of headers) 285
 - Expect header 273
 - GET request 271
 - headers 271, 290
 - sending as separate message 285
 - protocol 271
 - handling 284
 - PUT request 272
 - reply 271–272, 285
 - request 271
 - general form of 284
 - request line 284
 - resource 273
 - status code 272
 - verbs 272
 - HTTP client, curl 271
 - HTTP server
 - and gws_server 281
 - creating 270
 - handling requests 289
 - http_bin 279, 284
 - http_eoh 285
 - http_interface 287
 - htpd, Inets 274
 - hyperthreads 5
-
- I
 - I 73
 - i() 26
 - I/O system 19
 - usage, measuring during profiling 361
 - IC 295
 - idempotent, definition 233
 - identifier 35
 - IDL Compiler (IC). *See* IC
 - IEEE 754-1985 30
 - if expression 58
 - ifdef 74
 - ifndef 74
 - include 73
 - declarations 43
 - directive 73
 - directory 121
 - path 73
 - include_lib 73
 - inconsistent state, temporary 217
 - index, in Mnesia 231
 - inet:setopts/2 281
 - Inets httpd 274
 - inets httpd 198
 - infinity 77, 128, 161
 - info_msg 182
 - info_report 182
 - init 77
 - init_tables() 225
 - init:stop() 26
 - in remote shell 202
 - init:stop/0 116
 - init/1 108, 111
 - initial call 99
 - initialization code, location 165
 - inlining 368
 - insert/2 268
 - instrument (profiling tool) 361
 - instrument module 363
 - instrumenting code 187
 - integer 29
 - arbitrary size 29
 - bignums 370
 - base-N notation 29
 - division 30
 - no smallest 88
 - range of small integers 369
 - specifier in bitstring 68
 - to-float conversion, automatic 30
 - integration testing 17, 117
 - interactive development 53
 - interactive mode 252
 - interpreting 140
 - io module 41
 - io_lib:fwrite/2 116
 - io:format(...) 52
 - io:format/1 11, 173
 - io:format/2 173–174
 - IO-list 293
 - io-list 115
 - IP address, and start_link 277
 - IP packet header parsing 68
 - IPsec 198
 - is_atom/1 55
 - is_boolean/1 55
 - is_integer/1 55
 - isolation 227
 - of processes 7

- iteration 82
- I-vars 10
- J**

- j command 202
- jar file 344
- Java
 - and Jinterface 334
 - and plain ports 295
 - and list length 91
 - array vs. ETS table 80
 - classes in Jinterface to represent Erlang data 335
 - CLASSPATH 354
 - cp flag 339
 - decomposing data
 - structures 349
 - Erlang node 340
 - futures 10
 - Java node 292
 - looping 82
 - main() 338–339, 348
 - memory management 19
 - message handling 346–349
 - example 337–339
 - node, talking to from
 - Erlang 339–340
 - process() 338
 - program, compiling 337
 - run() 349
 - strings, constant on 381
 - thread pool 342, 347
 - threads 346
 - managing 334
- java file 337
- java_src 354
- java_src directory 344
- java.util.concurrent 348
- javac 337
- Javadoc vs. EDoc 101
- JavaScript Object Notation. *See* JSON
- Jinterface 292, 334–340
 - EPMD 339
 - Java classes to represent
 - Erlang data 335
 - mailboxes 334
 - OtpErlangObject 335–336
 - OtpMbox 334, 346
 - OtpNode 334, 338, 347
- JInterfaceExample.java 337
- job
 - connecting to 202
 - killing 27
- job control 27–28, 202
 - remote jobs 203
- JSON 291
 - data, encoding as Erlang terms 307–312
 - parser
 - implementing as NIF 322–331
 - integrating with through plain port 296–313
 - running 303–304
 - representation in Erlang 307
- JVM 9
- K**

- Kegel, Dan 19
- key/value pair 151
- kill 76, 136
- killing a process from
 - Appmon 136
- L**

- lambda expressions 36, 59, 61
- LAN, home, nodes on 196
- large integer
 - bignum 370
 - size 369
- last call 84
- latency, and synchronous
 - cache 218
- Latin-1 29, 31
- LD_LIBRARY_PATH 312
- lease time 158, 169
- left arrow 66
- length, avoiding in guards 91
- length/1 373
- let it crash 115, 240
- lib 255
- library application 120, 274
 - vs. active application 121
- LINE 73
- linear time 89, 91
- link request 78
- linked-in driver 293, 295
 - creating 313–322
 - driver callback functions 316
 - erl_driver.h 315
 - instance-specific data 314
 - lifecycle functions 316
 - managing memory 317
 - reentrant code 313
 - safety vs. speed 295
- sending output from port 319
- underlying mechanism 313–314
- Linux
 - installing Erlang on 379–380
 - starting the Erlang shell 24
- Lisp 31, 33
- list 33–34
 - adding to 33
 - adding to the end 382
 - adding to the left 34, 90, 382
 - and referential transparency 382
 - appending 34
 - cells 382
 - representation 372
 - checking if empty 91
 - deep list 115
 - empty 33
 - head 38
 - improper 39
 - io-list 115
 - length of 91
 - point of 381
 - proper 39
 - representation of 372
 - reversing 34, 86
 - singly linked 38
 - size 369
 - storage requirements 372
 - structure 38
 - tail 38
 - understanding 38–39
- list cell 51, 87
 - creating list from 38
- list comprehension 66, 229
- list_to_atom/1 373
- list_to_existing_atom/1 371, 373
- lists module 39, 41
- lists:append/2 373
- lists:delete/2 208
- lists:foldl/3 211
- lists:foreach/2 200
- lists:reverse/1 40
- lists:reverse/2 40
- lists:sort/1 37
- lists:subtract/2 373
- little, specifier in bitstring 68
- load balancer, stateless 213
- local call 40
- local flag 250
- location transparency 17, 193, 201, 219
- location-transparent syntax 192

- lock, definition 8
 - locking, problems with 9
 - log format 179
 - log4j 171, 348, 354
 - logging 171–179
 - error reports 176
 - example 181
 - format 179
 - report type 174
 - severity levels 171–172
 - standard functions 173–175
 - long jump 62
 - long node name 334
 - lookup/1 268
 - loop
 - ensuring termination 88
 - forever 86
 - initialization 85
 - initializing 83
 - looping 82
 - ls() 26, 44
 - ls(Dir) 26
- M**
-
- Mac OS X
 - Homebrew 379
 - installing Erlang on 379–380
 - starting the Erlang shell 24
 - machine words 369
 - macro
 - defining 72
 - expanding 72
 - MODULE 155, 158, 163
 - predefined 72
 - SERVER 155, 158
 - TABLE_ID 163–164
 - undefining 72
 - magic cookie 198–199
 - mailbox 7, 12
 - creating 335
 - extracting messages 76
 - ordering of messages 76
 - sending and receiving
 - messages 335
 - mailing list 92
 - main function 307
 - Make 129
 - make_ref() 36, 343
 - make_tar() 253
 - mapping
 - indirect 152
 - with list comprehension 66
 - MapReduce 20
 - marshalling, not needed 210
 - master process 245
 - match operator 47, 49
 - match specification 229
 - math 30
 - math:pow/2 66
 - math:sqrt/1 30
 - memory
 - dynamically allocated 314
 - region 20
 - shared 7, 16
 - usage, measuring during
 - profiling 361
 - memory management 19
 - and referential
 - transparency 382
 - memory() 26
 - message 97
 - decomposition in Java vs.
 - Erlang 349
 - delivery 78
 - naming 161
 - out-of-band 112–114
 - receiving 76
 - sending asynchronously 106
 - sending to a process on a different machine 193
 - sending to a process on the same machine 192
 - verification 218
 - message handling, in Java 337–339
 - message passing 7–8, 10–11, 192, 199–201
 - asynchronous 10
 - basics 12
 - by copying 20
 - synchronous 10
 - message queue, length of 136
 - messages, broadcasting 210
 - messaging 105
 - meta-call 116, 374
 - metadata 125, 243
 - MIB 122
 - MinGW 300
 - Mnesia 219–230
 - and HBase 333
 - cache, distributing 230
 - connecting to nodes and replicating data 240
 - database
 - creating 220–221
 - initializing 221–223
 - default settings 224
 - dirty operations 227
 - dirty write 232
 - distributing 238
 - indexes 231, 233
 - initializing depending on
 - nodes discovered 239
 - instead of ETS 231
 - name 220
 - node, starting 221
 - queries 228–230
 - reading 228
 - record, writing 227
 - records 221, 224
 - resource discovery,
 - integrating 236–238
 - schema 222
 - in RAM 239
 - remote, fetching 240
 - searching 228
 - starting 222
 - storage types 225
 - switching to, from ETS 230–233
 - tables
 - creating 223–226
 - dynamic replication 238
 - populating 226–228
 - storage types 225
 - types 224
 - viewing in TV 144
 - transactions 226
 - using QLC 229
 - mnesia, index_read/3 233
 - mnesia:change_config/2 240
 - mnesia:create_schema/1 222
 - mnesia:create_table/2 223
 - mnesia:delete_schema/1 222
 - mnesia:dirty_write/1 227
 - mnesia:info() 222
 - mnesia:read/2 228
 - mnesia:select/2 228
 - mnesia:start() 222
 - mnesia:system_info(tables) 240
 - mnesia:table/1 230
 - mnesia:transaction/1 227, 229
 - mnesia:wait_for_tables/2 240
 - MochiWeb 198, 274
 - mod parameter 123
 - mode 252
 - MODULE 72, 108
 - module 39–46, 96
 - calling a function in 40
 - canonical behavior
 - implementation 100
 - compiled vs. evaluation in shell 45
 - compiling 43–44

module (*continued*)
 compiling from the shell 43
 creating 42–43
 declaration 42
 dependencies 128
 loading, automatic 44
 name prefix 101
 namespace 101
 naming conventions 152
 one process type per 105
 Module:handle_call/3 104
 Module:handle_cast/2 104
 Module:init/1 104, 110
 modules parameter 123
 modulo operator 30
 monads 21
 monitor 76
 monitor/2 76
 monotonically decreasing
 arguments 88
 most specific pattern 92
 multicore 5
 MultiLisp, futures 10
 multithreading, and referential
 transparency 382
 mutex 8
 M-vars 10
 my_module 44, 86
 my_module.erl 42, 81, 83

N

-name 194, 334
 named_table 80, 164
 native implemented function.
See NIF
 native, specifier in bitstring
 68
 NavigableMap 345
 nc 271
 net_adm:ping/1 196, 199,
 234–235
 netcat 271
 network
 communicating over 192
 fully connected 193
 network byte order 302
 NIF 292, 295–296
 erl_nif API 322, 324
 ERL_NIF_INIT 326
 erl_nif.h 323
 erlang:nif_error/1 323
 ErlNifEnv 324
 implementation C
 functions 326

implementing 325
 implementing parser as
 322–331
 managing memory 324
 -on_load(...) attribute 323
 problems with 296
 registering 326
 stub functions 323
 nil 33
 creating list from 38
 nocatch 62
 node 193–203
 clusters 193
 joining 233–236
 communication 197
 connecting 195–197
 contact 234
 controlling remotely
 201–203
 cookies and 198
 discovery 197
 hidden 195
 implementing with
 Jinterface 334
 Java, talking to from
 Erlang 339–340
 locating other nodes 195
 long names 194, 334
 message passing 199–201
 network (cluster) 193
 remote shell job 202
 setting a cooking when
 creating 334
 short names 194, 334
 starting 194–195
 subnet 194
 temporary 203
 node() 194
 Nodefinder 197
 nodes() 196
 nonlocal return 62
 nonode@nohost 194
 noreply 162
 normal 162
 -noshell 24
 Not Invented Here
 syndrome 270

O

object file, introduction 44
 ok 32
 -on_load(...) attribute 323
 one_for_one 127
 one_for_one supervision 155

one_for_one supervisor 186
 open source Erlang 92
 Open Telecom Platform. *See*
 OTP
 open_port/2 297–298
 OpenSSH, and HBase 341
 operating system process 18
 operation, dirty, in Mnesia 227
 optimization. *See* performance
 tuning
 or 54
 order-constrained tasks 5
 ordered set 224
 ordsets:subtract/2 373
 OTP 4
 application 120–125
 metadata 122–123
 organization of 120–122
 development team 92
 OTP behaviour 376
 OTP Test Server 117
 OtpErlang.jar 337
 OtpErlangObject 335–336
 OtpMbox 334
 receive() 336
 OtpNode 334
 own time 365
 Oz, futures 10

P

-pa 129, 249
 {packet, http_bin} 279, 284
 pang 196
 parallelism 5
 limited by bottleneck 361
 parameter, accumulator
 parameter 85
 parser 269
 PATH 45
 pattern 50–52
 most specific 92
 pattern matching 33, 49–50,
 349
 and clause selection 54
 checking if a list is empty 91
 equality operator and 37
 in list comprehensions 66
 selecting clause with 56
 with bit syntax 68
 with record syntax 71
 peer-to-peer system 204
 percept 360–361
 performance goals 358

- performance tuning 358
 - baselines for
 - measurements 359
 - cprof 361–363
 - deciding which problems to
 - attack 360
 - determining performance
 - goals 358
 - fprof 363–368
 - measuring results 360
 - profiling the system 359
 - steps 358
- period character 54
 - in shell 24
- permanent 128, 162, 245
- pid. *See* process identifier
- ping
 - (command-line tool) 196
 - periodically reconnecting 212
- pizza, delivery 273
- planetlang.org 92
- Pman 137–140
 - in Toolbar 146
 - menus 138
 - Trace window 139
- pman:start() 137
- pong 196
- port 36, 292–295
 - binary option 298
 - communicating with 293, 298–299
 - concurrency control 299
 - exit_status option 298–299, 321
 - external program 293
 - in C 300–312
 - format 36
 - linked-in driver. *See* linked-in driver
 - opening 297–298
 - owner 294, 297
 - {packet,N} option 298, 302, 321
 - plain 293–295
 - speed 295
 - port ID 293
 - size 369
 - standard I/O streams 293
- port 4369 197
- port driver. *See* linked-in driver
- posting events 185
- preprocessor 72–74
 - conditionals 74
 - include files 73
 - macros 72
- pretty-printing 174
- primary key 144
- printable character 35
- printing text 11, 52
- priv directory 122, 298, 344
- proc_lib 179
- proc_lib:hibernate/3 378
- proc_lib:spawn/1 178
- procedural code 82
- process 5, 74–79, 97
 - as memory region 377
 - as storage element 219
 - child 12
 - communication 7, 12
 - by copying 7
 - paradigms 8–11
 - creating 11
 - default heap size 377
 - definition 6
 - efficiency of 376–378
 - encapsulating state 7
 - ending by throwing an
 - exception 76
 - example of 7
 - exit signal 13
 - trapping 14
 - hibernating 377–378
 - identifier 12
 - initial call 99
 - isolation 7
 - layering 15–16
 - lightweight 11
 - links 13
 - mailbox 7, 12, 105
 - monitoring 76
 - operating on 75–76
 - programming with 11–13
 - protocol 106
 - registered 77
 - registered on other node 199
 - restarting 15
 - scheduler 18–19
 - scheduling 366
 - sending explicit signal to 76
 - setting initial heap size 377
 - singleton 77, 107
 - size of 138
 - spawn signature 99
 - spawning 11
 - stack 84
 - stack space 11
 - supervisor process 14
 - suspended 77
 - suspension 366
 - system process 14
 - terminating 12
 - trapping exit signals 15
 - type 99
 - worker process 14
- process communication 78
 - by copying 191–192
 - by message passing 192
 - through shared memory 191
- process dictionary 79
- process flag 14
- process identifier 35, 193
 - deciphering 200
 - format 35
 - size 369
 - uniqueness of 35
- process mailbox, size of 137
- Process Manager. *See* Pman
- process model 6–8
- process queue 133
- process tree 125
- process_flag/2 76
- producer, in resource
 - discovery 204
- productivity
 - and OTP 4
 - due to task separation 6
- profile_ex 361, 363
- profile.txt 364
- profiling 360–368
 - cprof 361–363
 - example code 361
 - fprof 363, 368
 - measuring 361
 - where the most time is spent 360
 - tools 361
- profiling, during performance
 - tuning 359
- Programming Erlang—Software for a Concurrent World* 92
- programming resources 92
- Prolog
 - less-than-or-equals 37
 - syntax 21
- promises 10
- protocol message 208
- protocol, documenting 185
- public 164
- PUT request 272
- put/2 79
- PuTTY 116
- pwd() 26

Q

q() 26
 beware in remote shell 202
 QLC 229
 qlc:q(...) 229
 quadratic time 89, 91, 373
 avoiding 90
 Query List Comprehension. *See*
 QLC

R

r command 202
 Rake 129
 ram_copies 225, 239
 re 115
 readability
 and referential
 transparency 382
 due to task separation 6
 receive 12, 76
 blocking 77
 selective 76
 traversal of mailbox 77
 record 69–71
 in Mnesia 224
 record syntax 33
 record_info/2 224
 recursion 81–92
 body recursion 84
 over numbers 88
 tail recursion 84
 tail vs. body, efficiency of 375
 recursive call 84
 recursive case 86, 88
 simplest non-base case 90
 recursive function 82
 tips for writing 86
 reduction 138
 reentrant code 313
 ref. *See* reference
 reference 36
 size 369
 reference-counted binary 371
 referential transparency 33, 61,
 381
 advantages 381
 and lists 382
 region, in memory 20
 register/2 78
 registered parameter 123
 registered() 77
 regular expression 115
 .rel file 247, 255

release 242
 basics 246
 configuration 250–251
 creating 245–252
 installing 256–257
 interface apps, including
 in 263
 metadata 247–249
 packaging 253–256
 preparing for 247
 root path 257
 versioning 246
 release atom 248
 release file 247–249
 release package
 contents 254–255
 creating 253–254
 customizing 255–256
 structure of 254
 release_handler 256
 reliability, and OTP 4
 rem 30, 66
 remainder operator 30
 remote call 40
 remote monitoring 188
 remote procedure call. *See* RPC
 remote shell 201–203
 remove_handler 181
 replace/2 161
 representational state transfer.
 See REST
 resource 205
 resource discovery 204–212
 algorithm 205
 broadcasting messages 210
 dynamic 204
 fetching information 209–211
 implementation 206–212
 integrating 236–238
 making an application of 236
 module header 207
 publishing simple_cache 237
 starting the server 207–209
 terminology 204
 resource trading 209
 triggering 211
 resource tuple 205
 resource, HTTP 273
 REST 286
 DELETE 286
 GET 286
 PUT 287
 restart frequency 127
 RESTful 286
 interface, implementation
 of 287
 reuseaddr 279
 rev/1 86
 reversing a list 86
 RFC 2222 171
 root supervisor 120, 123–125,
 134, 152, 243
 implementing 155–157
 round-trip time, multiplied 218
 RPC
 vs. remote call 40
 RPC server 122
 implementing 100–116
 running 116–117
 Runnable 349
 runtime error 62
 runtime_tools 136, 363

S

s command 202
 SASL 249, 256
 crash report 178–179
 event handler 177
 extending logging with 172
 introduction 171
 log messages 176
 logging and 175, 179
 OTP vs. network
 authentication 171
 starting 176
 when it doesn't help 178
 sc_prefix 152
 sc_app 152, 155
 sc_element 152
 API section 158–160
 gen_server callback
 section 160–162
 header 157
 processes, coding 157–162
 sc_element_sup 186
 sc_element:replace/2 168
 sc_element:terminate/2 166
 sc_hbase 342
 sc_store 152, 231
 data, reading 165
 entries 165–166
 implementing 162–167
 initializing 164–165
 sc_store:delete(Pid) 162
 sc_store:init/0 238
 sc_store:lookup/1 168
 sc_sup 152, 156, 186

- scalability 16
 - and referential transparency 382
- scheduler 18–19
- schema 222
- SCons 129
- scope 47
 - of variables, in function clauses 56
- .script file 249
- selective receive 105
- self() 36
- semaphore 8
- send and pray 10, 214
- SERVER 108, 156
- server timeout 161
- server, and recursion 84
- set 81, 224, 228
 - ordered 224
- set notation 65
- set_cookie/2 199
- shallow copying 71
- shared library 320, 330
- shared memory 7–9
 - use in process communication 191
- sharing, in concurrent systems 8
- shell 23–28
 - compiling a module from 43
 - ending expression with period 24
 - entering strings 25
 - escaping from 26–27
 - interpreted code 45
 - introduction 23
 - job, killing 28
 - process 36
 - remote shell 201
 - starting 24
 - variables 47
- shell functions 26
- shell prompt 24, 202
 - node name in 195
- short node name 334
- shutdown 128
 - brutal_kill 156
 - soft 128
- signal 78
- signals, propagation of 13
- signed, specifier in bitstring 68
- Simple Cache
 - application-specific events 185
 - bridge to HBase 342–355
 - Erlang code 342–344
 - HBaseConnector class 344–346
 - HBaseNode class 346–349
 - HBaseTask class 349–351
 - custom event stream 183–189
 - event stream
 - integrating 185–187
 - subscribing to 188–189
 - interface with HBase 336
 - logging 171
- simple_cache 152
 - API module, creating 167–169
 - exported functions 268
- simple_cache:delete/1 168, 353
- simple_cache:insert/2 168, 352
- simple_cache:lookup/1 168, 352
- simple_cache.app 344
- simple_cache.erl 351
- simple_one_for_one restart strategy 185
- simple_one_for_one supervision 155, 159
- simple-one-for-one supervisor, gen_server, managing 262
- single assignment 47–49, 381
- single point of failure 16
- single quotes 32
- single-stepping, in debugger 141
- singleton process 107
- size/1 374
- small integer 369
- SMART 358
- SMP 18
- sname 194, 334
- .so 320
- socket
 - active 111, 113
 - active mode 267–268
 - blocking on accept() 267
 - dedicated 266
 - listening 111, 266
 - opening 264
 - supervising 266
 - listening, ownership of 264
 - message from 114
 - writing data 115
- soft shutdown 128
- software transactional memory. *See* STM
- sorting problem 5
- source code 42
- source file 42, 122
- spawn 11–12, 75, 361
- spawn signature 99
- spawn_link 75
- spawn_opt 75, 377
- src directory 122
- SSH 198
- SSH server, required for HBase 341
- sshd 341
- SSL 198
- stability, and OTP 4
- stability, and referential transparency 382
- stack 84
- stack trace 64
- stackoverflow.com 92
- standard library 41
- start 99
- start_link 99, 156, 277
- start_link() 184
- start_link/0 107
- start_link/1 107
- start.boot 255
- state
 - encapsulating 7
 - inconsistent, temporary 217
- statistics system 188
- stdlib 120
 - proc_lib 179
- STM 8–9
- stop 112, 162, 181
- stop/0 107
- storage element, process as 219
- storage, decoupling from application 163
- string 34
 - as list 34
 - constant, in Java 381
 - entering in shell 25
 - in binary syntax 31
 - in the shell 35
 - matching on prefixes 51
 - memory usage 372
 - prefixes, matching with ++ 51
 - printable characters in 35
- struct 33
- subnet, nodes 194
- subsystem termination 14
- succeed or die 234
- sum/1 81
- supervision
 - and OTP 4
 - one_for_one 155
 - simple_one_for_one 155, 159

- supervision structure,
 - viewing 134
 - supervision tree 15–16, 125, 127
 - dynamically generated 156
 - supervisor 152, 185, 187
 - adding fault tolerance with 125–129
 - avoid adding code to 264
 - avoiding application code in 280
 - child specification 127–128
 - children 156
 - creating 125
 - implementing 126–127
 - introduction 119, 125
 - module 156
 - naming 186
 - nesting 187
 - restart strategy 127, 155
 - root 126
 - self-termination 127
 - simple-one-for-one, as
 - factory 266
 - temporary worker
 - process 156
 - supervisor behaviour 125, 138
 - supervisor hierarchy,
 - visualizing 132
 - supervisor process 14
 - supervisor_bridge 125
 - supervisor:start_child/2 156, 159
 - supervisor:start_link/3 126
 - symbols (Lisp) 31
 - synaptic 379
 - synchronization bottleneck 361
 - synchronous cache. *See* distributed cache, asynchronous
 - synchronous
 - communication 214–216
 - support for timeouts 216
 - synchronous request 107
 - syntax 21
 - sys.config 250, 255
 - system
 - configuration 250–251
 - profiling, during performance tuning 359
 - System Architecture Support
 - Libraries. *See* SASL
 - system load 133
 - system process 14
 - systools 249, 253
 - systools:make_script 249, 253
 - systools:make_tar/2 253
 - systools:script2boot/1 250
- T**
-
- Table Viewer. *See* TV
 - TABLE_ID macro 164
 - table, distributed 219
 - tail call 84
 - tail recursion
 - efficiency of 85, 375
 - to avoid quadratic time 90
 - vs. non-tail recursion 85
 - tail, of a list 38
 - tailrev 90
 - .tar 253
 - tar 253
 - tarball 253
 - target system 246
 - definition 242
 - starting 251–252
 - tasks
 - order-constrained 5
 - separating 6
 - TCP 95
 - connecting via 116
 - connection, accepting 264
 - inspecting traffic over 271
 - listening socket 111
 - TCP port, and start_link 277
 - TCP server
 - efficient servers in Erlang 262–263
 - highly concurrent, pattern for 263
 - implementing 262–270
 - TCP socket
 - {active, once} 281
 - active mode 279
 - built-in HTTP parsing 279, 284
 - flow control 279
 - options 278
 - ownership of listening socket 279
 - {packet, raw} 285
 - passive mode 279
 - tcp_interface 263
 - directory structure 264
 - telnet 116, 270
 - temporary 128, 245
 - term 29
 - comparing 36
 - in ETS table 79
 - pretty-printing 53
 - printing 53
 - term_to_binary/1 299, 305, 343
 - terminate/2 callback 162
 - test case, writing 117
 - test, in guards 55
 - test-driven development 86
 - testing 117–118
 - text interface 268–270
 - text, printing 52
 - text-based protocol,
 - implementing 267–268
 - .tgz 253
 - Thompson, Simon 92
 - thread pool 342
 - threads 11, 18
 - communicating via messages 334
 - concurrency 7
 - stack space 11
 - throughput 359
 - throw 62
 - throw/1 62
 - ti_app 264–265
 - ti_server 266–267
 - handle_data/3 268
 - OTP-compliant factory for 266
 - ti_sup 265–266
 - time
 - accumulated time 365
 - changing default measurement 365
 - CPU time 360
 - own time 365
 - wall-clock time 360, 365
 - timeout 77, 113, 216
 - after init/1 111
 - timeout message 161
 - timing, nondeterministic 19
 - token 72
 - tokenizer 269
 - Toolbar 146
 - toolbar:start() 146
 - tools application 363
 - tr_server:start_link() 128
 - tracing
 - a warning 139
 - fprof built on top of 363
 - in Appmon 136
 - in Pman 139
 - trade_resources() 209
 - traffic, tunneling 198
 - transaction 9
 - in Mnesia 226
 - transient 128, 245
 - transparent distribution 8

trap_exit 14, 76
 trapexit.org 92
 traversing a data structure 86
 true 32

- and Boolean function 54
- in case expressions 57

 try 62, 115

- after 64
- body 63
- try...of 63

 try/catch, handling errors

- in 269

 tunneling Erlang traffic 198
 tuple 32

- access/update times 372
- boxed representation 370
- creation, fast 71
- inflexibility of 69
- pattern matching 33
- record syntax 33
- representation of 372
- size 369
- tagged 33, 56, 70

 tuple_size/1 374
 TV 144–145

- in Toolbar 146
- viewing tables 144

 tv:start() 144
 two-phase commit 218
 type specifications, in EDoc

- comments 109

 type test 55
 TypeSpecifiers 68

U

Ubuntu, synaptic 379
 undef 72
 undefined 32, 70
 underscore 32

- as don't-care pattern 51
- starting variable name with 46

 Unicode 29
 unit test 86
 unit testing 117

UNIX

- installing Erlang on 379–380
- starting the Erlang shell 24

 unsigned, specifier in

- bitstring 68

 unused-variable warning 46
 upgradability, and OTP 4
 User Switch Command

- menu 27

 UTF encoding, in bitstrings 68
 utf16 68
 utf32 68
 utf8 68

V

v(N) 24, 26
 variable 46–52

- and referential transparency 381
- binding 47, 49
- external 313
- in the shell 47
- inspecting in debugger 141
- multiple occurrences in patterns 50
- scope, in function clauses 56
- single assignment 47–50
- starting with underscore 46
- syntax 46
- updates 48

 vector comparison 37
 verification message 218
 version string 123
 versioning, of releases 246
 Viriding, Robert 92
 virtual machine emulator 17–20
 vsn 244
 vsn parameter 123

W

wait_for_nodes/2 236
 wall-clock time 365
 warning_msg 174, 182

warning_report 182
 weak consistency 217
 web server 7

- as container 274
- implementing as generic behaviour 274–286
- logging in 214

 web service, building 270–290
 WebAppmon 136
 WebDAV 277
 WebTool 136, 257
 webtool:start() 136
 werl 24, 27, 45, 129, 379
 when 55
 whereis/1 77
 while loop 85
 Wikström, Claes 92
 Williams, Mike 92
 Windows 27

- erlc compiler in 45
- installer 379
- installing Erlang on 379
- starting the Erlang shell 24
- werl shell 379

 wireless LAN, and distributed mode 194
 worker 128, 187
 worker process 14

Y

YAJS 291

- handling key/value pairs 309
- managing memory 317
- parser callbacks 307–312, 324, 327
- running 303–304

 yajl_alloc 304
 yajl_free 304
 yajl_free_error 304
 yajl_get_error 304
 yajl_parse_complete 304
 Yaws 193, 198, 274
 YECC 122