Table des matières

Préface xvii
Avant-propos xix

Bases du langage, spécificités de Lua..... 1

1. Informations générales 3
1. Installer Lua 3
2. Sur quels systèmes puis-je installer Lua ?..... 3
3. Quel éditeur pour développer en Lua ?..... 3
4. Comment Lua se situe-t-il par rapport aux autres langages en termes de performances ?..... 4
5. Quelles versions faut-il utiliser ?..... 4
6. Quels outils pour documenter son code ?..... 4
7. Quels sont les sites de référence pour développer en Lua ?..... 4
8. Comment passer de Lua 5.1 à Lua 5.2 ?..... 5

2. Principes et éléments de syntaxe 7
9. De quoi est composé Lua ?..... 7
10. Les commentaires 8
11. Les commentaires sur plusieurs lignes 8
12. Les commentaires d'affichage 8
13. Les instructions 9
14. Les mots clés du langage 9
15. Afficher des résultats 9
16. Comment la mémoire est-elle allouée ou libérée ?..... 10
17. Comment connaître la version de Lua utilisée ?..... 10

3. Variables et types 11
18. Quel est le type d'une variable ?..... 11
19. Quels sont les types de données disponibles ?..... 11
20. Le type *nil* 11
21. Le type *boolean* 12
22. Le type numérique 12
23. Le type *string* 12
24. Le type *function* 13
25. Le type *userdata* 13
26. Qu'est-ce qu'un *thread* Lua ?..... 13

27. Le type *table* 13
28. Un type peut-il être converti ?..... 13
29. Connaître le type d'une donnée 14
30. Quelle est la portée d'une variable ?..... 14

4. Expressions et opérateurs 15
31. Quels sont les opérateurs arithmétiques ?..... 15
32. Quels sont les opérateurs relationnels ?..... 15
33. Quels sont les opérateurs logiques disponibles ?..... 16
34. Autres opérateurs à connaître 17

5. Portée des variables, blocs et chunks..... 19
35. Quand utiliser *local* ?..... 21
36. Où sont stockées les variables globales ?..... 21

6. Structures de contrôle 23
37. Comment déclarer un bloc de code ?..... 23
38. Comment exprimer des conditions ?..... 23
39. Les expressions ternaires existent-elles comme en C/C++ ? 24
40. Comment faire un *switch/case* ?..... 24
41. Comment effectuer des boucles ?..... 24
42. Utiliser les boucles de type *while* 25
43. Utiliser les boucles de type *repeat* 25
44. Utiliser les boucles de type *for* 25
45. Créer une boucle *for* numérique 26
46. Créer une boucle *for* générique 27
47. Comment sortir d'une boucle ?..... 28
48. Peut-on faire un *try/catch* comme en Java ou C++ ? 28
49. Existe-il une instruction *continue* ?..... 29
50. Effectuer un saut inconditionnel avec *goto* 29

7. Fonctions 31
51. Nombre inconsistant d'arguments et de valeurs 32
52. Nommer des arguments pour les sélectionner lors de l'appel..... 32
53. Déclarer une liste variable d'arguments 33
54. Renvoyer plusieurs résultats 34
55. Que sont les fonctions anonymes ?..... 34
56. Que sont les *closures* ?..... 34

8. Tables 37
57. Construire une table et y accéder 37
58. Initialiser une table 38

59. Créer et utiliser un enregistrement 39
60. Comment obtenir la taille d'une table ?..... 39
61. Pourquoi une indexation numérique à partir de 1 ?..... 40
62. Table de fonctions 40
63. Instancier des objets avec les tables 41
64. Parcourir une table avec un *for* générique 42
65. Copier une table 43
66. Que sont les *weak tables* ?..... 43
67. Utiliser des tables pour faire des matrices 44
68. Le module *table* 44
69. La fonction *table.concat* 45
70. Insérer une valeur dans une séquence 45
71. Supprimer une valeur dans une séquence 45
72. Créer un tableau depuis une liste de valeurs 46
73. Extraire une liste de valeurs d'une séquence 46
74. Trier une séquence 46

9. Les fonctions internes 49
75. *assert()* 49
76. *collectgarbage()* 50
77. *dofile()* 52
78. *error()* 52
79. *setmetatable()/getmetatable()* 52
80. *ipairs()/pairs()* ?..... 52
81. *next()* 53
82. *loadfile()/load()* 53
83. *pcall()/xpcall()* 53
84. *print()* 54
85. *rawequal()/rawget()/rawlen()/rawset()* 54
86. *require()* 54
87. *select()* 54
88. *tonumber()* 55
89. *tostring()* 55
90. *type()* 55

10. La gestion des erreurs 57
91. Quelles sont les erreurs générées ?..... 57
92. Remonter des erreurs 57

11. Les coroutines 59
93. Créer une coroutine 59

94. Connaître l'état des coroutines 61
95. Que fait la fonction *coroutine.wrap()* ?..... 62

Lua, librairies et modules 65

12. Appeler et exécuter du code externe..... 67
1. Charger et exécuter un script 67
2. Charger un script à partir d'une chaîne de caractères 68
3. Charger un script à partir d'un fichier 69
4. Créer une librairie 70
5. Gérer un fichier de configuration 70
6. Contrôler le code chargé par les fonctions *load()/loadfile()* 71

13. Créer ses librairies 73
7. Le module *package* 73
8. Appeler une librairie avec *require()* 73
9. Comment les modules sont-ils recherchés ?..... 73
10. Changer les chemins de recherche 75
11. La variable *package.config* 75
12. Écrire un module Lua destiné à être chargé par *require()* 76
13. Écrire un module C destiné à être chargé par *require()* 78
14. Accéder depuis Lua à une librairie C ou C++..... 78
15. Déclarer le *loader* d'une librairie 79
16. À quoi sert *package.loadlib()* ?..... 79

14. Les métatables 81
17. Quels opérateurs peut-on modifier avec les métatables ?..... 81
18. Peut-on ajouter de nouveaux opérateurs aux métatables ?..... 82
19. À quoi cela peut-il servir ?..... 82
20. Utiliser la métaméthode multiplication (*__mul*) 85
21. Utiliser la métaméthode division (*__div*) 86
22. Utiliser la métaméthode modulo (*__mod*) 86
23. Utiliser la métaméthode de négation (*__unm*) 87
24. Utiliser la mise à la puissance (*__pow*) 87
25. Utiliser la métaméthode de concaténation 88
26. Utiliser les métaméthodes de comparaison 88
27. Utiliser la métaméthode de ramasse-miettes 89
28. Accéder à un indice avec *__index* 90
29. Simuler une classe avec des méthodes 90
30. Créer un index avec *__newindex* 91

31. Retourner la taille d'une valeur avec __*len* 92
32. Capturer un appel de fonction avec __*call* 92
33. Modifier les fonctions *pairs()* et *ipairs()* 92
34. Éviter l'appel à une métaméthode 92
35. Peut-on parler de classe comme en C++ ? 93
36. Peut-on interdire la modification des métatables ?..... 93
37. Comment supprimer une métatable ?..... 93

Manipuler ses données et ses fichiers..... 95

15. Les chaînes de caractères 97
1. Déclarer des chaînes 98
2. Déclarer des chaînes sur plusieurs lignes 99
3. Concaténer des chaînes 99
4. Obtenir la taille d'une chaîne 100
5. Conversions automatiques 100
6. Le module intégré *string* 100
7. Formater des chaînes 101
8. Extraire des portions de chaînes 101
9. Convertir en minuscules ou majuscules 102
10. Récupérer les codes de caractères d'une chaîne 102
11. Créer une chaîne à partir de codes de caractères 103
12. Répéter une chaîne 103
13. Inverser une chaîne 103
14. Transformer une fonction en chaîne 104
15. Et l'Unicode ?..... 104

16. Recherche de motifs dans des chaînes (*pattern matching*) 105
16. Manipuler des chaînes à l'aide de motifs 105
17. Les classes de caractères dans les motifs 106
18. Construire des motifs complets 108
19. Extraire des portions de chaînes 109
20. Rechercher un motif, avec indices 109
21. Rechercher simplement un motif 110
22. Itérer sur la recherche d'un motif 110
23. Remplacer un motif 111

17. La librairie LPeg 115
24. Utiliser LPeg 115
25. Vérifier un motif 116

26. Déclarer un motif 117
lpeg.P 117
lpeg.S et *lpeg.R* 117
27. Utiliser des jeux de caractères prédéfinis 118
28. Préciser le nombre d'occurrences 119
29. Combiner des motifs 120
30. Rechercher un motif n'importe où dans une chaîne 121
31. Exemple de traitement avec LPeg d'une expression régulière complexe 122
32. Extraire des séquences 125
33. Capturer un motif simple 125
34. Capturer la position du caractère suivant le motif 126
35. Capturer un motif n'importe où dans une chaîne 126
36. Insérer des valeurs constantes 127
37. Retourner un argument de *lpeg.match()* 128
38. Appliquer une fonction à la capture 128
39. Grouper les valeurs capturées 129
40. Récupérer les valeurs du dernier groupe nommé 129
41. Récupérer les captures dans une table 130
42. Combiner des captures 131
43. Que signifie l'expression *motif/valeur* ?..... 133
44. Remplacer la capture par une valeur 135
45. Capturer en temps réel 136
46. Créer une grammaire 138
47. Comment mettre au point les motifs ?..... 141
48. Le module RE 147

18. Calculs mathématiques 149
49. Tronquer ou arrondir des nombres 149
50. Fonctions trigonométriques 150
51. Fonctions de conversions angulaires 150
52. Fonctions logarithmiques et exponentielles 150
53. Conversion de fractions normalisées 151
54. Mettre à la puissance et calculer des racines carrées 151
55. Obtenir des minimum et maximum 151
56. Les constantes mathématiques 152
57. Calculer la valeur absolue 152
58. Générer des nombres aléatoires 152
59. Extraire une partie entière et fractionnaire 152
60. Calcul de modulo réel 153

19. Calculs logiques 155
61. Les opérations logiques usuelles 155
62. Test logique 156
63. Champs de bits 156
64. Décalages logiques 157
65. Rotations logiques 157
66. Décalage arithmétique 158

20. Gestion des fichiers 159
67. Généralités sur les entrées/sorties 159
68. Comment lire un fichier ?..... 160
69. Lire un petit fichier texte 160
70. Lire un petit fichier binaire 160
71. Peut-on faire une lecture plus fine ?..... 161
72. Lire un fichier texte ligne par ligne 162
73. Écrire un petit fichier 163
74. Mettre à jour un fichier 163
75. Ajouter des lignes à un fichier 164
76. Écrire un gros fichier 164
77. Le modèle simple de *io* 165

21. Le module LFS et ses utilisations 167
78. Lister les fichiers d'un répertoire 167
79. Lister les fichiers et les sous-répertoires d'un répertoire..... 168
80. Manipuler des répertoires 170
81. Manipuler des fichiers 170
82. Verrouiller répertoires et fichiers 171

S'interfacer avec le monde extérieur..... 173

22. Les fonctions d'interfaçage avec l'OS..... 175
1. Récupérer la date système avec *os.date()* 175
2. Récupérer la date système avec *os.time()* 176
3. Récupérer le temps CPU consommé 176
4. Calculer un écart de dates 176
5. Exécuter des commandes externes 176
6. Forcer la sortie d'un programme 177
7. Récupérer une variable d'environnement 177
8. Supprimer un fichier 178
9. Renommer un fichier 178

10. Changer les paramètres de localisation 178
11. Créer un fichier temporaire 179

23. Lua et *POSIX* 181
12. Quelles sont les fonctions disponibles ?..... 181
13. Quelles sont les fonctions *curses* disponibles ?..... 183
14. Comment utiliser *curses* ?..... 183

24. Les bases de données 185
15. Lua comme format de base de données 186
16. Exploiter des données CSV / TSV 188
17. Importer des fichiers JSON 190
18. Utiliser des fichiers YAML 191
19. Traiter du XML 191
20. Communiquer avec des bases SQL 192
21. LuaSQLite 3 195
22. Communiquer avec des bases NoSQL 196

25. Le réseau 197
23. *LuaSocket* 197
24. Créer un serveur TCP/IP 198
25. Créer un client TCP/IP 198
26. Envoyer et recevoir des données UDP 199
27. Effectuer des recherches DNS 200
28. Effectuer les opérations d'un client FTP 200
29. Le module *ltn12* 201
30. Récupérer le contenu d'une page web 201
31. Autres fonctions de *LuaSocket* 202

26. Les interfaces utilisateur graphiques 203
32. Qu'entend-on par interface utilisateur graphique ?..... 203
33. Quel système de GUI utiliser ?..... 204
34. Quelles liaisons avec Lua sont disponibles ?..... 205
35. Fonder sa GUI sur des composants natifs 205
36. Recourir à une boîte à outils de composants 206
37. Installer une librairie de type IUP, lqt ou tekUI 206
38. Écrire *Hello World!* avec IUP, lqt ou tekUI 207
39. Ajouter de l'interactivité avec IUP, lqt ou tekUI 209

27. Lua dans les jeux vidéo 211
40. Utiliser Lua dans le gameplay 211
41. Utiliser Lua pour les paramétrages du jeu 215

42. Utiliser Lua pour l'intelligence artificielle 216

S'interfacer avec le C 219

28. Les bases de l'API C..... 221
1. Changer la fonction d'allocation 222
2. Charger les modules de la librairie standard 223
3. Charger des scripts *via* l'API C..... 223
4. Opérations mathématiques depuis le C..... 224
5. Convertir une fonction en *chunk* 224

29. Manipulation de la pile d'appel 225
6. Modifier la pile 225
7. Comment savoir ce que contient la pile ?..... 226
8. Récupérer des variables globales 230
9. Comparer des données dans la pile 230
10. Concaténer des données dans la pile 231
11. Appeler une fonction 231

30. Manipulation des tables 235
12. Créer, modifier et lire une table 236
13. Parcourir une table avec l'API C..... 237
14. Utiliser les métatables depuis l'API C..... 238
15. Le registre Lua 239

31. Les fonctions C et les fermetures 241
16. Utiliser des fonctions C en Lua 241
17. Manipuler des fonctions C..... 242
18. Associer des valeurs à une fonction C..... 242
19. Manipuler les *upvalues* d'une fermeture 244

32. Les *userdata* 245
20. Qu'est-ce que les *userdata* ?..... 245
21. Qu'est-ce que les *lightuserdata* ?..... 245
22. Quand faut-il se servir des *userdata* ?..... 246
23. Créer une *userdata* 247
24. Échanger des *userdata* entre Lua et C 248
25. Associer une métatable à une *userdata* 248
26. Accès de type objet avec les *userdata* 250
27. Associer une table à une *userdata* 251
28. *userdata* et le ramasse-miettes ?..... 251

33. Utilisation avancée 253
29. Déboguer avec l'API C..... 253
30. Gérer les erreurs 253
31. Remonter une erreur 253
32. Gérer le ramasse-miettes 254
33. Créer et gérer des coroutines 254
34. Déplacer des données entre deux coroutines 256
35. Tester l'état de coroutines 257

34. La librairie auxiliaire 259
36. Quand utiliser la librairie auxiliaire ?..... 259
37. Créer un état Lua 261
38. Charger rapidement la librairie standard 261
39. Charger et exécuter un fichier Lua 261
40. Charger et exécuter une chaîne de caractères 262
41. Charger du code à partir d'un buffer 262
42. Vérifier et manipuler la pile d'appel 262
43. Facilité pour les énumérations C..... 264
44. Facilité pour les chaînes de caractères 264
45. Les tableaux et les métatables 264
46. Traiter les erreurs 265
47. Manipuler les buffers pour créer des chaînes de caractères 265
48. Vérifier dynamiquement la cohérence des versions..... 267
49. Facilité pour créer des librairies 268
50. Charger une librairie depuis le C..... 268
51. Obtenir une référence dans une table 268

Déboguer son code 269

35. Déboguer côté Lua 271
1. Quels outils pour déboguer du code Lua ?..... 271
2. Que peut-on faire avec le module *debug* ?..... 271
3. Débogage interactif 272
4. Obtenir des informations sur une fonction 272
5. Étudier ou modifier des variables locales 274
6. Étudier ou modifier des *upvalues* 275
7. Consulter et modifier la table associée à une *userdata* 276
8. Mettre en place une fonction d'hameçonnage 276
9. Afficher la pile d'appel 278

10. Manipuler les métatables 279
11. Récupérer le *registry* 279

36. Déboguer côté C 281
12. Comment déboguer ses scripts à partir du C ?..... 281
13. Obtenir des informations sur une fonction 281
14. Obtenir des informations sur la pile d'appel 282
15. Accéder à des variables locales 283
16. Accéder à des valeurs *upvalue* 283
17. Mettre en place une fonction d'hameçonnage 283

L'implémentation LuaJIT 285

37. L'implémentation LuaJIT 287
1. Quand LuaJIT est-il plus performant que Lua ?..... 287
2. Quelles sont les limitations de LuaJIT ?..... 288
3. Quelle compatibilité entre LuaJIT et les versions de Lua officielles ?..... 288
4. Quelles extensions apporte LuaJIT ?..... 288
5. Arrêter ou démarrer la compilation dynamique 289
6. Obtenir l'état de la compilation dynamique 289
7. Récupérer la version de LuaJIT 289
8. Récupérer des informations système 290

38. Le module *FFI* de LuaJIT 291
9. Quelle différence avec les mécanismes Lua ?..... 291
10. Principe d'utilisation de *FFI* 291
11. Déclarer les fonctions à importer 292
12. Comment les types sont-ils convertis ?..... 293
13. Comment résoudre les symboles déclarés avec *ffi.cdef()* ?..... 294
14. Peut-on déclarer des variables ou des types ?..... 295
15. Récupérer des informations sur la plate-forme 297
16. Peut-on utiliser des callbacks C ?..... 298
17. Peut-on associer des métatables au type *cdata* ?..... 300
18. Obtenir des informations sur les variables *cdata* 301
19. Récupérer la valeur de *errno* 302
20. Créer une *string* depuis un pointeur C 302
21. Copier le contenu de variables *cdata* ou *string* 302
22. Initialiser le contenu d'un type *cdata* 302

39. Le module *BitOp* de LuaJIT 303

23. Conversion en chaînes hexadécimales 303
24. Opérations logiques usuelles 303
25. Décalages logiques 304
26. Rotations logiques 304
27. Décalage arithmétique 305
28. Changer l'ordre des octets 305
29. Retourner une valeur normalisée 305

Annexe 307
Lexique 309

Index 319
À propos des auteurs 335