

Programmation avancée en C

avec exercices et corrigés

Sébastien Varrette
Nicolas Bernard

Table des matières

Avant-propos	17
Le langage C	18
Organisation de cet ouvrage	19
Notations	19
Remerciements	20
 Chapitre 1. Survol rapide du langage C	21
1.1. Un premier programme	21
1.1.1. Compilation	22
1.1.1.1. Que se passe-t-il à la compilation ?	22
1.1.1.2. La compilation en pratique	23
1.1.2. Éléments du programme <i>Hello World</i>	24
1.2. Étude d'un second exemple et structure d'un programme C typique . .	25
1.2.1. Directives au préprocesseur	25
1.2.2. Déclarations et définitions de variables	26
1.2.3. Définitions de fonctions	27
1.3. Notion d'identificateur	28
1.4. Conventions d'écritures d'un programme C	28
1.5. Les principaux types	28
1.5.1. Les caractères	29
1.5.2. Les entiers	29
1.5.3. Les flottants	29
1.5.4. Le type void	30
1.5.5. Les autres types	30
1.6. Structures de contrôle	30
1.7. Opérateurs et expressions	31
1.8. Entrées / sorties de base	32
1.8.1. La fonction d'écriture à l'écran <code>printf</code>	32
1.8.2. La fonction de saisie <code>scanf</code>	33

1.9. Exercices	33
PREMIÈRE PARTIE. LE LANGAGE C	35
Chapitre 2. Syntaxe et sémantique générales	37
2.1. Jeu de caractères	37
2.2. Les mots-clefs	38
2.3. Les commentaires	38
2.4. Expressions et Opérateurs	39
2.4.1. Les opérateurs arithmétiques	41
2.4.2. Les opérateurs bit-à-bit	42
2.4.3. Les opérateurs d'affectation	42
2.4.4. Les opérateurs relationnels	42
2.4.5. Les opérateurs logiques	43
2.4.6. Les opérateurs d'accès à la mémoire	44
2.4.7. Autres opérateurs	45
2.5. Constructions et structures de contrôle	46
2.5.1. Les branchements conditionnels	46
2.5.1.1. La construction <code>if...else</code>	46
2.5.1.2. La construction <code>switch</code>	47
2.5.2. Les boucles	49
2.5.2.1. La construction <code>for</code>	49
2.5.2.2. La construction <code>while</code>	50
2.5.2.3. La construction <code>do...while</code>	51
2.5.3. Les branchements inconditionnels	51
2.5.3.1. L'instruction <code>goto</code> et les étiquettes	52
2.5.3.2. L'instruction <code>break</code>	52
2.5.3.3. L'instruction <code>continue</code>	53
2.5.3.4. L'instruction <code>return</code>	53
2.5.4. Les autres constructions	53
2.5.4.1. La construction de regroupement	53
2.5.4.2. La <i>construction-expression</i> et l'instruction nulle	54
2.6. Fonctions et variables	54
2.6.1. Déclarations	54
2.6.2. Définitions (et appels) de fonctions	56
2.6.2.1. Prototype et corps d'une fonction	56
2.6.2.2. Une fonction particulière, <code>main</code>	57
2.6.2.3. Appel d'une fonction	57
2.6.2.4. Récursivité	57
2.6.2.5. Passage de paramètres	58
2.6.2.6. Fonctions à nombre variable de paramètres	60
2.6.3. Définitions et portée des variables	61
2.6.3.1. Les variables globales	61

2.6.3.2. Les variables locales	62
2.7. Exercices	63
Chapitre 3. Types	69
3.1. Les types entiers	69
3.1.1. Les caractères	69
3.1.2. Les entiers	71
3.1.2.1. Types à taille imposée ou minimale et types rapides de C99	71
3.1.2.2. Valeurs minimales et maximales	73
3.1.2.3. Cas des constantes entières	73
3.1.3. Nombres signés / non signés et leurs représentations	74
3.1.4. Les booléens	75
3.2. Les nombres à virgule flottante	76
3.2.1. Représentation interne	77
3.2.2. Cas des constantes flottantes	77
3.2.3. Les nombres complexes	78
3.3. Le type void	79
3.4. Spécificateurs de classe de stockage	80
3.5. Le spécificateur de fonction inline	81
3.6. Qualificatifs de types	82
3.6.1. const	82
3.6.2. volatile	83
3.6.3. restrict	84
3.7. Définition de synonymes pour les noms de types avec typedef	84
3.8. Les conversions de type	85
3.8.1. La règle de "promotion des entiers"	86
3.8.2. Les conversions arithmétiques habituelles	86
3.8.3. Les surprises de la conversion de type	86
3.9. Les pointeurs	87
3.9.1. Déclaration d'un pointeur	89
3.9.2. Les pointeurs génériques void *	89
3.9.3. Le pointeur spécial NULL	90
3.9.4. Opérateurs de manipulation des pointeurs	90
3.9.4.1. L'opérateur 'adresse de' &	90
3.9.4.2. L'opérateur 'contenu de l'adresse' *	91
3.9.5. Initialisation d'un pointeur	92
3.9.6. Arithmétique des pointeurs	93
3.9.7. Allocation et libération dynamique de mémoire	95
3.9.7.1. Allocation dynamique de mémoire avec malloc et calloc	95
3.9.7.2. Libération de mémoire avec free	97
3.9.7.3. Un mot sur les autres fonctions d'allocation dynamique	98
3.10. Pointeurs sur une fonction	99
3.11. Les énumérations	100

3.12. Les tableaux	101
3.12.1. Initialisation d'un tableau	102
3.12.2. Le cas particulier des chaînes de caractères	104
3.12.3. Tableaux multidimensionnels	106
3.12.4. Passage de tableau en paramètre	107
3.12.4.1. Modification des éléments d'un tableau passé en paramètre	108
3.12.4.2. Interdire la modification des éléments d'un tableau passé en paramètre	109
3.12.4.3. Passage d'une copie de tableau à une fonction	109
3.12.5. Relation entre tableaux et pointeurs	110
3.12.5.1. Pointeurs et tableaux à une dimension	110
3.12.5.2. Pointeurs et tableaux à plusieurs dimensions	112
3.12.6. Cas des tableaux de chaînes de caractères	113
3.12.7. Gestion des arguments de la ligne de commande	114
3.13. Les structures	117
3.13.1. Initialisation et affectation d'une structure	118
3.13.2. Comparaison de structures	118
3.13.3. Tableau de structures	118
3.13.4. Pointeur vers une structure	119
3.13.5. Structures auto-référées	119
3.13.5.1. Ajout d'éléments dans une liste chaînée	120
3.13.5.2. Suppression d'un élément dans une liste chaînée	122
3.13.5.3. Exemple d'utilisation de listes simplement chaînées	123
3.13.6. Les champs de bits	124
3.14. Les unions	125
3.14.1. Déclaration d'une union	125
3.14.2. Utilisation pratique des unions	126
3.14.3. Une méthode pour alléger l'accès aux membres	127
3.15. Les littéraux composés	127
3.16. Exercices	128

Chapitre 4. Entrées / Sorties 135

4.1. Entrée standard, sorties standards, et utilisation des formats	135
4.1.1. Lecture et écriture de caractères depuis stdin et stdout : <code>getchar</code> et <code>putchar</code>	136
4.1.2. Écriture de chaînes sur la sortie standard : <code>puts</code>	136
4.1.3. Lecture de chaînes depuis l'entrée standard : <code>gets</code>	136
4.1.4. Écriture formatée de chaînes sur la sortie standard : <code>printf</code>	137
4.1.5. Lecture formatée de chaînes depuis l'entrée standard : <code>scanf</code>	139
4.2. Ouverture et fermeture de flots de données	140
4.2.1. Ouverture de fichiers : la fonction <code>fopen</code>	140
4.2.2. Détection de fin de fichier	142
4.2.3. Fermeture de fichiers : la fonction <code>fclose</code>	142

4.3. Lectures/écritures sur flots de données	143
4.3.1. Lecture et écriture de caractères sur un flot de données : <code>fgetc</code> et <code>fputc</code>	143
4.3.2. Écriture de chaînes sur un flot de données : <code>fputs</code>	143
4.3.3. Lecture de chaînes depuis un flot de données : <code>fgets</code>	143
4.3.4. Écriture formatée de chaînes sur flot de données : <code>fprintf</code>	144
4.3.5. Lecture formatée de chaînes depuis un flot de données : <code>fscanf</code>	144
4.3.6. Lecture / écriture formatée sur chaîne de caractères : <code>sprintf</code> , <code>snprintf</code> et <code>sscanf</code>	144
4.3.7. Entrées-sorties binaires : <code>fread</code> et <code>fwrite</code>	145
4.3.8. Positionnement dans un flot de données	145
4.4. Exercices	146
Chapitre 5. Les directives du préprocesseur	149
5.1. Inclusion de fichiers : la directive <code>#include</code>	149
5.2. Définition de <i>macros</i>	150
5.2.1. Définition de constantes	150
5.2.1.1. Constantes symboliques prédéfinies	151
5.2.1.2. Définition de constantes symboliques à l'invocation du com- pilateur	151
5.2.2. Les macros avec paramètres	152
5.2.2.1. Erreurs fréquentes	153
5.2.2.2. Conversion et fusion de lexèmes	153
5.2.2.3. Comment abuser des macros : le Bourne shell	154
5.3. La compilation conditionnelle	154
5.3.1. Condition liée à la valeur d'une expression	155
5.3.2. Condition liée à l'existence d'un symbole	155
5.3.2.1. L'opérateur <code>defined</code>	156
5.3.3. La commande <code>#error</code>	157
5.4. Directives plus rarement utilisées	157
5.4.1. <code>#line</code>	157
5.4.2. La commande <code>#pragma</code>	157
5.4.2.1. Pragmas standards	158
5.4.2.2. L'opérateur <code>_Pragma</code>	159
5.4.3. La directive nulle	159
5.5. Exercices	159
Chapitre 6. La bibliothèque standard	161
6.1. Implémentations <i>hosted</i> / <i>freestanding</i>	161
6.2. Diagnostics d'erreurs : <code>assert.h</code>	162
6.3. Gestion des nombres complexes <code>complex.h</code>	162
6.4. Classification de caractères et changements de casse <code>ctype.h</code>	162
6.5. Numéro de la dernière erreur <code>errno.h</code>	162

6.6. Gestion de l'environnement à virgule flottante <code>fenv.h</code>	163
6.6.1. Gestion des exceptions	164
6.6.2. Gestion des arrondis	164
6.6.3. Gestion des environnements en virgule flottante.	164
6.6.4. Contraction des expressions	165
6.7. Intervalle et précision des nombres flottants <code>float.h</code>	165
6.8. Extension des types entiers <code>inttypes.h</code> et <code>stdint.h</code>	165
6.9. Alias d'opérateurs logiques et binaires <code>iso646.h</code>	166
6.10. Intervalle de valeur des types entiers <code>limits.h</code>	166
6.11. Gestion de l'environnement local <code>locale.h</code>	166
6.12. Les fonctions mathématiques de <code>math.h</code>	167
6.12.1. Gestion des erreurs	167
6.12.2. Fonctions trigonométriques et hyperboliques	168
6.12.3. Fonctions exponentielles, puissances et logarithmiques	168
6.12.4. Autres fonctions	168
6.12.5. Quelques ajouts de C99	168
6.13. Branchements non locaux <code>setjmp.h</code>	169
6.14. Manipulation des signaux <code>signal.h</code>	170
6.15. Nombre variable de paramètres <code>stdarg.h</code>	170
6.16. Définition du type booléen <code>stdbool.h</code>	170
6.17. Définitions standards <code>stddef.h</code>	171
6.18. Gestion des entrées / sorties <code>stdio.h</code>	171
6.18.1. Manipulation de fichiers	171
6.18.2. Lectures / écritures de base	172
6.18.3. Lectures / écritures formatées de chaînes de caractères	172
6.19. Utilitaires généraux <code>stdlib.h</code>	172
6.19.1. Allocation dynamique	172
6.19.2. Génération de nombres pseudo-aléatoires	172
6.19.3. Conversion de chaînes de caractères en nombres	173
6.19.4. Arithmétique sur les entiers	174
6.19.5. Recherche et tri dans un tableau	174
6.19.6. Contrôles d'exécution et d'environnement	175
6.20. Manipulation de chaînes de caractères <code>string.h</code>	175
6.21. Macros génériques pour les fonctions mathématiques <code>tgmath.h</code>	178
6.22. Date et heure <code>time.h</code>	178
6.23. Manipulation de caractères étendus <code>wchar.h</code> et <code>wctype.h</code>	180

DEUXIÈME PARTIE. PROGRAMMATION AVANCÉE EN LANGAGE C 181

Chapitre 7. Tester et débbugger un programme 183

7.1. Débbugger avec le langage lui-même	184
7.1.1. Afficher des valeurs : <code>printf</code>	184
7.1.2. <code>assert</code>	185

7.2. Débuggeurs	185
7.2.1. Gdb, le débogueur GNU	186
7.2.1.1. Utilisation des fichiers <i>core</i>	191
7.2.1.2. « Connexion » à un processus en cours d'exécution	192
7.2.1.3. Interfaces graphiques et extensions	192
7.2.2. Valgrind	193
7.3. Bibliothèques spécialisées	194
7.4. Tests	195
7.4.1. Jeux de tests	195
7.4.2. Fuzzing	196
7.5. Pour aller plus loin, d'autres outils	196
7.6. Exercices	197
Chapitre 8. Programmation modulaire et bibliothèques	199
8.1. Programmation modulaire	200
8.1.1. Éviter les erreurs d'inclusions multiples	201
8.1.2. La compilation séparée	201
8.1.3. Quelques notes supplémentaires	202
8.2. Bibliothèques	202
8.2.1. Bibliothèques statiques et partagées	202
8.2.2. Utiliser une bibliothèque	203
8.2.3. Créer une bibliothèque	204
8.2.4. Interposition	205
8.3. Introduction à l'outil make	207
8.3.1. Principe de base	208
8.3.2. Création d'un Makefile	208
8.3.3. Variables	210
8.3.3.1. Noms standards	210
8.3.3.2. Variables automatiques	211
8.3.3.3. Wildcards et substitutions	211
8.3.4. Règles génériques	212
8.3.5. Inclusions	212
8.3.6. Utiliser gcc pour générer une liste de dépendances	212
8.3.7. Nettoyage	213
8.4. Exercices	213
Chapitre 9. Quelques notions de systèmes et d'architecture	215
9.1. Système d'exploitation, mode noyau, mode utilisateur	215
9.2. Disposition d'un programme en mémoire	217
9.3. Appels de fonction	219
9.4. Appels système	222
9.5. « Boutisme »	223
9.6. Alignement des données	224

9.7. Caches	228
9.8. Exercices	228
Chapitre 10. Programmation en C et sécurité	231
10.1. Tout le monde est concerné	232
10.2. Failles de sécurité courantes en C	232
10.2.1. Débordements de tampons	232
10.2.1.1. Débordement sur la pile	232
10.2.1.2. Débordements sur le tas	234
10.2.1.3. « String-format vulnerabilities »	235
10.2.1.4. Prévention des débordements	236
10.2.2. Race conditions	237
10.2.3. Débordements sur les nombres	238
10.2.4. Problèmes de chaînes de caractères	241
10.2.4.1. Difficulté d'utilisation de <code>strncat</code> et <code>strncpy</code>	241
10.2.4.2. Une alternative non standard : <code>strlcat</code> et <code>strncpy</code>	242
10.2.4.3. Vérification de troncature de chaînes	243
10.2.5. Fichiers temporaires	244
10.3. Mieux vaut prévenir que guérir	244
10.3.1. Validation des entrées	244
10.3.2. Confinement et séparation des pouvoirs	245
10.3.3. Outils d'analyse de code	246
10.3.4. Cryptographie	247
10.4. Exercice	247
Chapitre 11. Optimisation	251
11.1. Mesures de performances	252
11.1.1. Ajouter du code pour mesurer le temps	252
11.1.2. Profilers	253
11.1.2.1. <code>gprof</code>	253
11.1.2.2. Valgrind : Cachegrind, Callgrind et Massif	256
11.2. Techniques d'optimisation	257
11.2.1. « Bonne programmation »	257
11.2.1.1. Le principe de localité	257
11.2.1.2. « micro-algorithmes », l'exemple de la copie en place	257
11.2.2. Trucs et astuces	258
11.2.3. Utiliser les possibilités du système et du compilateur	261
11.2.3.1. <code>gcc</code> et le C GNU	261
11.2.3.2. Les appels système <code>posix_madvise</code> et <code>posix_fadvise</code>	262
11.3. Exercices	262

Chapitre 12. Utiliser C avec d'autres langages	263
12.1. Inclusion de l'assembleur dans du C avec gcc	264
12.2. Interfacer C avec un autre langage	266
12.2.1. Un cas simple : C++	266
12.2.1.1. Utiliser du code C dans un programme C++	267
12.2.1.2. Utiliser du code C++ dans un programme C	268
12.2.1.3. Précautions	269
12.2.2. Swig	270
12.2.3. L'exemple de Perl	271
12.2.4. Utiliser du code d'un langage de haut niveau depuis C	275
12.2.5. Exercices	276
TROISIÈME PARTIE. INTRODUCTION À LA PROGRAMMATION POSIX	277
Chapitre 13. Programmation système Unix / POSIX	279
13.1. Accès aux fichiers	280
13.1.1. Les principaux appels système d'accès aux fichiers	280
13.1.2. Position dans un fichier	283
13.1.3. Descripteurs de fichiers standards	284
13.1.4. Duplication de descripteurs	284
13.1.5. Encore quelques appels sur les fichiers	285
13.2. Gestion de processus	285
13.2.1. Création de processus et exécution de programme	285
13.2.2. Attendre un processus	288
13.3. Communications et synchronisation inter-processus	289
13.3.1. Signaux	290
13.3.1.1. Envoyer des signaux	291
13.3.1.2. Définir une fonction à appeler lors de la réception d'un signal	291
13.3.1.3. Bloquer des signaux	292
13.3.2. Tubes	294
13.3.3. Mapping de fichiers, mémoire partagée et sémaphores	296
13.3.3.1. mmap	296
13.3.3.2. Sémaphores	300
13.4. Threads	301
13.4.1. Gestion des threads	302
13.4.2. Synchronisation des threads	302
13.5. Exercices	304
Chapitre 14. Sockets et programmation réseau POSIX	307
14.1. Quelques notions de réseaux	307
14.1.1. Modèles OSI et TCP/IP	308
14.1.2. Encapsulation des données	308
14.1.3. Modèle de communication Client / Serveur	309

14.1.3.1. Définition	309
14.1.4. Adressage réseau	310
14.1.4.1. Représentation en C d'une adresse IPv4	311
14.1.4.2. Représentation en C d'une adresse IPv6	311
14.1.4.3. Gestion de l'ordre des octets sur réseau et sur machine . . .	311
14.1.4.4. Manipulation des adresses Internet	312
14.1.5. Ports et services réseaux	315
14.1.6. Nommage	317
14.2. Généralités sur les sockets	324
14.2.1. Caractéristiques	324
14.2.1.1. Domaines de communication	324
14.2.1.2. Types et protocoles supportés	325
14.2.2. Descripteurs de fichiers et descripteurs de sockets	326
14.2.3. Les différentes structures de socket	326
14.2.3.1. Structure de socket IPv4 <code>sockaddr_in</code>	326
14.2.3.2. Structure de socket IPv6 <code>sockaddr_in6</code>	327
14.2.3.3. Structure de socket Unix <code>sockaddr_un</code>	327
14.2.3.4. Structures génériques pour les adresses	327
14.2.3.5. Bilan sur les structures d'adresses	328
14.2.4. Fonctions communes sur les sockets	329
14.2.4.1. Création d'une socket : la fonction <code>socket</code>	329
14.2.4.2. Lier une adresse (locale) à une socket : <code>bind</code>	329
14.2.4.3. Récupérer l'adresse locale d'une socket : <code>getsockname</code> .	331
14.2.4.4. Récupérer l'adresse distante d'une socket : <code>getpeername</code> .	331
14.2.4.5. Fermeture d'une socket : <code>close</code>	332
14.3. Sockets TCP	332
14.3.1. Connexion TCP d'un client à un serveur via <code>connect</code>	333
14.3.2. Envoi et réception de données	333
14.3.3. Exemple d'un client TCP simple	334
14.3.4. Attente de connexions (sur le serveur) : <code>listen</code>	336
14.3.5. Accepter une connexion sur le serveur via <code>accept</code>	336
14.3.6. Traitement concurrentiel des requêtes clientes sur le serveur . . .	337
14.3.7. Fermeture paramétrable d'une socket TCP : <code>shutdown</code>	339
14.3.8. Exemple d'un serveur TCP simple	339
14.4. Sockets UDP	342
14.4.1. Envoi de données : <code>sendto</code>	343
14.4.2. Réception de données : <code>recvfrom</code>	343
14.4.3. Exemple d'un client / serveur UDP simple	344
14.5. Sockets Unix	346
14.5.1. Calcul de la taille effective d'une adresse de socket Unix	346
14.5.2. Socket Unix par flot	347
14.5.3. Socket Unix par datagrammes	350
14.5.4. Socket Unix anonymes : <code>socketpair</code>	350

14.6. Aspects avancés de la programmation des sockets	350
14.6.1. Multiplexage d'entrées / sorties avec <code>select</code>	350
14.6.2. Entrées / sorties non bloquantes	352
14.6.3. Passage de descripteurs entre processus	352
14.7. Exercices	354
Annexes	355
A. Correction des exercices	355
B. Licences	403
B.1. Licence du <code>stdbool.h</code> d'OpenBSD	403
B.2. Licence du <code>gets</code> d'OpenBSD	403
B.3. Licence du <code>exec1</code> d'OpenBSD	404
Bibliographie	405
Index	409