

Le guide complet du langage **C**

Claude Delannoy



EYROLLES

© Groupe Eyrolles, 1999, 2008, 2014, ISBN : 978-2-212-14012-5

Table des matières

Avant-propos	1
À qui s'adresse ce livre ?	1
Structure de l'ouvrage	2
À propos des normes ANSI/ISO	4
À propos de la fonction main	4
Remerciements	5
 CHAPITRE 1	
Généralités	7
1. Historique du langage C	7
2. Programme source, module objet et programme exécutable	8
3. Compilation en C : existence d'un préprocesseur	9
4. Variable et objet	10
4.1 Définition d'une variable et d'un objet	10
4.2 Utilisation d'un objet	10
5. Lien entre objet, octets et caractères	12
6. Classe d'allocation des variables	12
 CHAPITRE 2	
Les éléments constitutifs d'un programme source	15
1. Jeu de caractères source et jeu de caractères d'exécution	16
1.1 Généralités	16
1.2 Commentaires à propos du jeu de caractères source	17
1.3 Commentaires à propos du jeu minimal de caractères d'exécution	18

2. Les identificateurs	19
3. Les mots-clés	20
4. Les séparateurs et les espaces blancs	20
5. Le format libre	21
6. Les commentaires	22
7. Notion de token	23
7.1 Les différentes catégories de tokens	23
7.2 Décomposition en tokens	25
CHAPITRE 3	
Les types de base	27
1. Les types entiers	27
1.1 Les six types entiers	28
1.2 Représentation mémoire des entiers et limitations	29
1.3 Critères de choix d'un type entier	32
1.4 Écriture des constantes entières	33
1.5 Le type attribué par le compilateur aux constantes entières	34
1.6 Exemple d'utilisation déraisonnable de constantes hexadécimales	35
1.7 Pour imposer un type aux constantes entières	36
1.8 En cas de dépassement de capacité dans l'écriture des constantes entières	36
2. Les types caractère	37
2.1 Les deux types caractère	37
2.2 Caractéristiques des types caractère	38
2.3 Écriture des constantes caractère	40
2.4 Le type des constantes caractère	43
3. Le fichier limits.h	45
3.1 Son contenu	45
3.2 Précautions d'utilisation	46
4. Les types flottants	46
4.1 Rappels concernant le codage des nombres en flottant	46
4.2 Le modèle proposé par la norme	47
4.3 Les caractéristiques du codage en flottant	48
4.4 Représentation mémoire et limitations	50
4.5 Écriture des constantes flottantes	51

4.6 Le type des constantes flottantes	52
4.7 En cas de dépassement de capacité dans l'écriture des constantes	52
5. Le fichier float.h	53
6. Déclarations des variables d'un type de base	54
6.1 Rôle d'une déclaration	55
6.2 Initialisation lors de la déclaration	56
6.3 Les qualifieurs const et volatile	57
CHAPITRE 4	
Les opérateurs et les expressions	61
1. Généralités	62
1.1 Les particularités des opérateurs et des expressions en C	62
1.2 Priorité et associativité	63
1.3 Pluralité	64
1.4 Conversions implicites	64
1.5 Les différentes catégories d'opérateurs	65
2. Les opérateurs arithmétiques	65
2.1 Les différents opérateurs numériques	66
2.2 Comportement en cas d'exception	69
3. Les conversions numériques implicites	74
3.1 Introduction	74
3.2 Les conversions numériques d'ajustement de type	75
3.3 Les promotions numériques	80
3.4 Combinaisons de conversions	86
3.5 Cas particulier des arguments d'une fonction	88
4. Les opérateurs relationnels	89
4.1 Généralités	89
4.2 Les six opérateurs relationnels du langage C	90
4.3 Leur priorité et leur associativité	91
5. Les opérateurs logiques	93
5.1 Généralités	93
5.2 Les trois opérateurs logiques du langage C	94
5.3 Leur priorité et leur associativité	95
5.4 Les opérandes de && et de ne sont évalués que si nécessaire	96

6. Les opérateurs de manipulation de bits	97
6.1 Présentation des opérateurs de manipulation de bits	97
6.2 Les opérateurs « bit à bit »	98
6.3 Les opérateurs de décalage.	100
6.4 Applications usuelles des opérateurs de manipulation de bits	102
7. Les opérateurs d'affectation et d'incrément	104
7.1 Généralités	104
7.2 La lvalue	105
7.3 L'opérateur d'affectation simple	106
7.4 Tableau récapitulatif : l'opérateur d'affectation simple.	108
7.5 Les opérateurs d'affectation élargie	109
8. Les opérateurs de cast	110
8.1 Généralités	110
8.2 Les opérateurs de cast	111
9. Le rôle des conversions numériques	113
9.1 Conversion d'un type flottant vers un autre type flottant	113
9.2 Conversion d'un type flottant vers un type entier.	114
9.3 Conversion d'un type entier vers un type flottant.	114
9.4 Conversion d'un type entier vers un autre type entier	115
9.5 Cas particuliers des conversions d'entier vers caractère.	116
9.6 Tableau récapitulatif des conversions numériques	118
10. L'opérateur conditionnel	119
10.1 Introduction.	119
10.2 Rôle de l'opérateur conditionnel	120
10.3 Contraintes et conversions	120
10.4 La priorité de l'opérateur conditionnel	122
11. L'opérateur séquentiel	122
12. L'opérateur sizeof	124
12.1 L'opérateur sizeof appliqué à un nom de type	124
12.2 L'opérateur sizeof appliqué à une expression	125
13. Tableau récapitulatif : priorités et associativité des opérateurs.	127
14. Les expressions constantes.	128
14.1 Introduction.	128
14.2 Les expressions constantes d'une manière générale.	129

CHAPITRE 5

Les instructions exécutables	133
1. Généralités	133
1.1 Rappels sur les instructions de contrôle	134
1.2 Classification des instructions exécutables du langage C	134
2. L'instruction expression	136
2.1 Syntaxe et rôle	136
2.2 Commentaires	136
3. L'instruction composée ou bloc	137
3.1 Syntaxe d'un bloc	137
3.2 Commentaires	138
3.3 Déclarations dans un bloc	139
3.4 Cas des branchements à l'intérieur d'un bloc	140
4. L'instruction if	140
4.1 Syntaxe et rôle de l'instruction if	140
4.2 Exemples d'utilisation	141
4.3 Cas des if imbriqués	144
4.4 Traduction de choix en cascade	145
5. L'instruction switch	147
5.1 Exemple introductif	147
5.2 Syntaxe usuelle et rôle de switch	148
5.3 Commentaires	149
5.4 Quelques curiosités de l'instruction switch	150
6. Choix entre if et switch	152
7. Les particularités des boucles en C	153
7.1 Rappels concernant la programmation structurée	153
7.2 Les boucles en C	154
8. L'instruction do ... while	155
8.1 Syntaxe	155
8.2 Rôle	156
8.3 Exemples d'utilisation	157
9. L'instruction while	158
9.1 Syntaxe	159
9.2 Rôle	159

9.3 Lien entre while et do ... while	160
9.4 Exemples d'utilisation.	161
10. L'instruction for	162
10.1 Introduction.	162
10.2 Syntaxe	163
10.3 Rôle.	163
10.4 Lien entre for et while.	164
10.5 Commentaires.	165
10.6 Exemples d'utilisation.	166
11. Conseils d'utilisation des différents types de boucles	168
11.1 Boucle définie	168
11.2 Boucle indéfinie	169
12. L'instruction break.	170
12.1 syntaxe et rôle.	170
12.2 Exemple d'utilisation	170
12.3 Commentaires.	171
13. L'instruction continue	172
13.1 Syntaxe et rôle.	172
13.2 Exemples d'utilisation.	172
13.3 Commentaires.	173
14. Quelques schémas de boucles utiles	175
14.1 Boucle à sortie intermédiaire	175
14.2 Boucles à sorties multiples	178
15. L'instruction goto et les étiquettes.	179
15.1 Les étiquettes	179
15.2 Syntaxe et rôle	180
15.3 Exemples et commentaires	181

CHAPITRE 6

Les tableaux.	185
1. Exemple introductif d'utilisation d'un tableau	186
2. Déclaration des tableaux	187
2.1 Généralités	187
2.2 Le type des éléments d'un tableau	188

2.3 Déclarateur de tableau	188
2.4 La dimension d'un tableau	190
2.5 Classe de mémorisation associée à la déclaration d'un tableau	192
2.6 Les qualifieurs const et volatile	193
2.7 Nom de type correspondant à un tableau	194
3. Utilisation d'un tableau	195
3.1 Les indices	195
3.2 Un identificateur de tableau n'est pas une lvalue	196
3.3 Utilisation d'un élément d'un tableau	196
3.4 L'opérateur sizeof et les tableaux	197
4. Arrangement d'un tableau et débordement d'indice.	198
4.1 Les éléments d'un tableau sont alloués de manière consécutive	198
4.2 Aucun contrôle n'est effectué sur la valeur de l'indice	199
5. Cas des tableaux de tableaux.	200
5.1 Déclaration des tableaux à deux indices	200
5.2 Utilisation d'un tableau à deux indices	200
5.3 Peut-on parler de lignes et de colonnes d'un tableau à deux indices ?	202
5.4 Arrangement en mémoire d'un tableau à deux indices	202
5.5 Cas des tableaux à plus de deux indices	203
6. Initialisation de tableaux	204
6.1 Initialisation par défaut des tableaux	205
6.2 Initialisation explicite des tableaux	205
CHAPITRE 7	
Les pointeurs	211
1. Introduction à la notion de pointeur	212
1.1 Attribuer une valeur à une variable de type pointeur	212
1.2 L'opérateur * pour manipuler un objet pointé	213
2. Déclaration des variables de type pointeur	214
2.1 Généralités	215
2.2 Le type des objets désignés par un pointeur	216
2.3 Déclarateur de pointeur	216
2.4 Classe de mémorisation associée à la déclaration d'un pointeur	219
2.5 Les qualifieurs const et volatile	219
2.6 Nom de type correspondant à un pointeur	223

3. Les propriétés des pointeurs	224
3.1 Les propriétés arithmétiques des pointeurs	225
3.2 Lien entre pointeurs et tableaux	226
3.3 Ordre des pointeurs et ordre des adresses	232
3.4 Les restrictions imposées à l'arithmétique des pointeurs	232
4. Tableaux récapitulatifs : les opérateurs +, -, &, * et []	235
5. Le pointeur NULL	237
6. Pointeurs et affectation	239
6.1 Prise en compte des qualifieurs des objets pointés	239
6.2 Les autres possibilités d'affectation	241
6.3 Tableau récapitulatif	242
6.4 Les affectations élargies += et -= et les incrémentations ++ et --	242
7. Les pointeurs génériques	243
7.1 Généralités	243
7.2 Déclaration du type void *	244
7.3 Interdictions propres au type void *	244
7.4 Possibilités propres au type void *	245
8. Comparaisons de pointeurs	246
8.1 Comparaisons basées sur un ordre	247
8.2 Comparaisons d'égalité ou d'inégalité	248
8.3 Récapitulatif : les comparaisons dans un contexte pointeur	249
9. Conversions de pointeurs par cast.	249
9.1 Conversion d'un pointeur en un pointeur d'un autre type	250
9.2 Conversions entre entiers et pointeurs	252
9.3 Récapitulatif concernant l'opérateur de cast dans un contexte pointeur	253

CHAPITRE 8

Les fonctions	255
1. Les fonctions en C	256
1.1 Une seule sorte de module en C : la fonction	256
1.2 Fonction et transmission des arguments par valeur	258
1.3 Les variables globales	258
1.4 Les possibilités de compilation séparée	259
2. Exemple introductif de la notion de fonction en langage C	259

3. Définition d'une fonction	261
3.1 Les deux formes de l'en-tête	262
3.2 Les arguments apparaissant dans l'en-tête	262
3.3 La valeur de retour	264
3.4 Classe de mémorisation d'une fonction : extern et static	268
3.5 L'instruction return	269
4. Déclaration et appel d'une fonction	272
4.1 Déclaration sous forme de prototype	273
4.2 Déclaration partielle (déconseillée)	274
4.3 Portée d'une déclaration de fonction	275
4.4 Redéclaration d'une fonction	276
4.5 Une définition de fonction tient lieu de déclaration	277
4.6 En cas d'absence de déclaration	278
4.7 Utilisation de la déclaration dans la traduction d'un appel.	278
4.8 En cas de non-concordance entre arguments muets et arguments effectifs	281
4.9 Les fichiers en-tête standards	282
4.10 Nom de type correspondant à une fonction	283
5. Le mécanisme de transmission d'arguments	284
5.1 Cas où la transmission par valeur est satisfaisante	284
5.2 Cas où la transmission par valeur n'est plus satisfaisante.	285
5.3 Comment simuler une transmission par adresse avec des pointeurs	287
6. Cas des tableaux transmis en arguments	289
6.1 Règles générales	289
6.2 Exemples d'applications	294
6.3 Pour qu'une fonction dispose de la dimension d'un tableau	296
6.4 Quelques conseils de style à propos des tableaux en argument	298
7. Cas particulier des tableaux de tableaux transmis en arguments	299
7.1 Application des règles générales	300
7.2 Artifices facilitant la manipulation de tableaux de dimensions variables	304
8. Les variables globales	310
8.1 Exemples introductifs d'utilisation de variables globales	311
8.2 Les déclarations des variables globales	313
8.3 Portée des variables globales	319
8.4 Variables globales et édition de liens	320
8.5 Les variables globales sont de classe statique	321
8.6 Initialisation des variables globales	322

9. Les variables locales	324
9.1 La portée des variables locales	324
9.2 Classe d'allocation et initialisation des variables locales	326
10. Tableau récapitulatif : portée, accès et classe d'allocation des variables	331
11. Pointeurs sur des fonctions	332
11.1 Déclaration d'une variable pointeur sur une fonction	333
11.2 Affectation de valeurs à une variable pointeur sur une fonction	334
11.3 Appel d'une fonction par le biais d'un pointeur	337
11.4 Exemple de paramétrage d'appel de fonctions	339
11.5 Transmission de fonction en argument	341
11.6 Comparaisons de pointeurs sur des fonctions	342
11.7 Conversions par cast de pointeurs sur des fonctions	342

CHAPITRE 9

Les entrées-sorties standards	345
1. Caractéristiques générales des entrées-sorties standards.	346
1.1 Mode d'interaction avec l'utilisateur	346
1.2 Formatage des informations échangées	347
1.3 Généralisation aux fichiers de type texte	347
2. Présentation générale de printf	348
2.1 Notions de format d'entrée, de code de format et de code de conversion	349
2.2 L'appel de printf	350
2.3 Les risques d'erreurs dans la rédaction du format	352
3. Les principales possibilités de formatage de printf	355
3.1 Le gabarit d'affichage	355
3.2 Précision des informations flottantes	358
3.3 Justification des informations	360
3.4 Gabarit ou précision variable	360
3.5 Le code de format g	362
3.6 Le drapeau + force la présence d'un signe « plus »	365
3.7 Le drapeau espace force la présence d'un espace	366
3.8 Le drapeau 0 permet d'afficher des zéros de remplissage	366
3.9 Le paramètre de précision permet de limiter l'affichage des chaînes	367
3.10 Cas particulier du type unsigned short int : le modificateur h	368

4. Description des codes de format des fonctions de la famille printf	369
4.1 Structure générale d'un code de format	369
4.2 Le paramètre drapeaux.	369
4.3 Le paramètre de gabarit	370
4.4 Le paramètre de précision	372
4.5 Le paramètre modificateur h/l/L	373
4.6 Les codes de conversion	373
4.7 Les codes utilisables avec un type donné.	375
5. La fonction putchar	376
5.1 Prototype.	377
5.2 L'argument de putchar est de type int.	377
5.3 La valeur de retour de putchar	378
6. Présentation générale de scanf	378
6.1 Format de sortie, code de format et code de conversion	378
6.2 L'appel de scanf	379
6.3 Les risques d'erreurs dans la rédaction du format	380
6.4 La fonction scanf utilise un tampon.	383
6.5 Notion de caractère invalide et d'arrêt prématuré	384
6.6 La valeur de retour de scanf.	385
6.7 Exemples de rencontre de caractères invalides.	387
7. Les principales possibilités de scanf	389
7.1 La présentation des informations lues en données	390
7.2 Limitation du gabarit	391
7.3 La fin de ligne joue un rôle ambigu : séparateur ou caractère	392
7.4 Lorsque le format impose certains caractères dans les données	394
7.5 Attention au faux gabarit du code C.	394
7.6 Les codes de format de la forme %[...]	395
8. Description des codes de format des fonctions de la famille de scanf	397
8.1 Récapitulatif des règles utilisées par ces fonctions	397
8.2 Structure générale d'un code de format	398
8.3 Les paramètres * et gabarit	399
8.4 Le paramètre modificateur h/l/L	399
8.5 Les codes de conversion.	400
8.6 Les codes utilisables avec un type donné.	402
8.7 Les différences entre les codes de format en entrée et en sortie	403

9. La fonction getchar	404
9.1 Prototype et valeur de retour	404
9.2 Précautions	404

CHAPITRE 10

Les chaînes de caractères	407
1. Règles générales d'écriture des constantes chaîne	408
1.1 Notation des constantes chaîne	408
1.2 Concaténation des constantes chaîne adjacentes	409
2. Propriétés des constantes chaîne	410
2.1 Conventions de représentation	411
2.2 Emplacement mémoire	412
2.3 Cas des chaînes identiques	413
2.4 Les risques de modification des constantes chaîne	413
2.5 Simulation d'un tableau de constantes chaîne	415
3. Créer, utiliser ou modifier une chaîne	416
3.1 Comment disposer d'un emplacement pour y ranger une chaîne	417
3.2 Comment agir sur le contenu d'une chaîne	418
3.3 Comment utiliser une chaîne existante	421
4. Entrées-sorties standards de chaînes	423
4.1 Généralités	423
4.2 Écriture de chaînes avec puts	424
4.3 Écriture de chaînes avec le code de format %s de printf ou fprintf	425
4.4 Lecture de chaînes avec gets	426
4.5 Lecture de chaînes avec le code de format %s dans scanf ou fscanf	429
4.6 Comparaison entre gets et scanf dans les lectures de chaînes	430
4.7 Limitation de la longueur des chaînes lues sur l'entrée standard	431
5. Généralités concernant les fonctions de manipulation de chaînes	435
5.1 Ces fonctions travaillent toujours sur des adresses	435
5.2 Les adresses sont toujours de type char *	435
5.3 Certains arguments sont déclarés const, d'autres pas	436
5.4 Attention aux valeurs des arguments de limitation de longueur	437
5.5 La fonction strlen	438

6. Les fonctions de copie de chaînes	439
6.1 Généralités	439
6.2 La fonction strcpy	439
6.3 La fonction strncpy	443
7. Les fonctions de concaténation de chaînes	446
7.1 Généralités	446
7.2 La fonction strcat	446
7.3 La fonction strncat	449
8. Les fonctions de comparaison de chaînes	452
8.1 Généralités	452
8.2 La fonction strcmp	453
8.3 La fonction strncmp	454
9. Les fonctions de recherche dans une chaîne	454
9.1 Les fonctions de recherche d'un caractère : strchr et strrchr	455
9.2 La fonction de recherche d'une sous-chaîne : strstr	458
9.3 La fonction de recherche d'un caractère parmi plusieurs : strpbrk	458
9.4 Les fonctions de recherche d'un préfixe	459
9.5 La fonction d'éclatement d'une chaîne : strtok	461
10. Les fonctions de conversion d'une chaîne en un nombre	464
10.1 Généralités	464
10.2 La fonction de conversion d'une chaîne en un double : strtod	465
10.3 Les fonctions de conversion d'une chaîne en entier : strtol et strtoul	469
10.4 Cas particulier des fonctions atof, atoi et atol	475
11. Les fonctions de manipulation de suites d'octets	475
11.1 Généralités	476
11.2 Les fonctions de recopie de suites d'octets	476
11.3 La fonction memcmp de comparaison de deux suites d'octets	477
11.4 La fonction memset d'initialisation d'une suite d'octets	478
11.5 La fonction memchr de recherche d'une valeur dans une suite d'octets	479
 CHAPITRE 11	
Les types structure, union et énumération	481
1. Exemples introductifs	482
1.1 Exemple d'utilisation d'une structure	482
1.2 Exemple d'utilisation d'une union	484

2. La déclaration des structures et des unions	486
2.1 Définition conseillée d'un type structure ou union	486
2.2 Déclaration de variables utilisant des types structure ou union.	490
2.3 Déclaration partielle ou déclaration anticipée	492
2.4 Mixage entre définition et déclaration	493
2.5 L'espace de noms des identificateurs de champs	494
2.6 L'espace de noms des identificateurs de types	495
3. Représentation en mémoire d'une structure ou d'une union	495
3.1 Contraintes générales	496
3.2 Cas des structures.	497
3.3 Cas des unions	498
3.4 L'opérateur sizeof appliqué aux structures ou aux unions	499
4. Utilisation d'objets de type structure ou union	499
4.1 Manipulation individuelle des différents champs d'une structure ou d'une union.	500
4.2 Affectation globale entre structures ou unions de même type.	501
4.3 L'opérateur & appliqué aux structures ou aux unions	503
4.4 Comparaison entre pointeurs sur des champs	503
4.5 Comparaison des structures ou des unions par == ou != impossible.	504
4.6 L'opérateur ->	504
4.7 Structure ou union transmise en argument ou en valeur de retour.	505
5. Exemples d'objets utilisant des structures	508
5.1 Structures comportant des tableaux	508
5.2 Structures comportant d'autres structures	509
5.3 Tableaux de structures	510
5.4 Structure comportant des pointeurs sur des structures de son propre type	511
6. Initialisation de structures ou d'unions	511
6.1 Initialisation par défaut des structures ou des unions.	512
6.2 Initialisation explicite des structures	513
6.3 L'initialisation explicite d'une union	517
7. Les champs de bits	517
7.1 Introduction.	517
7.2 Exemples introductifs	518
7.3 Les champs de bits d'une manière générale	519
7.4 Exemple d'utilisation d'une structure de champs de bits dans une union	522

8. Les énumérations	523
8.1 Exemples introductifs	523
8.2 Déclarations associées aux énumérations.	526
CHAPITRE 12	
La définition de synonymes avec typedef	529
1. Exemples introductifs	530
1.1 Définition d'un synonyme de int.	530
1.2 Définition d'un synonyme de int *	530
1.3 Définition d'un synonyme de int[3]	531
1.4 Définition d'un synonyme d'un type structure	532
2. L'instruction typedef d'une manière générale	532
2.1 Syntaxe	532
2.2 Définition de plusieurs synonymes.	533
2.3 Imbrication des définitions de synonyme.	534
3. Utilisation de synonymes	534
3.1 Un synonyme peut s'utiliser comme spécificateur de type.	534
3.2 Un synonyme n'est pas un nouveau type	535
3.3 Un synonyme peut s'utiliser à la place d'un nom de type	535
4. Les limitations de l'instruction typedef	536
4.1 Limitations liées à la syntaxe de typedef	536
4.2 Cas des tableaux sans dimension	537
4.3 Cas des synonymes de type fonction	538
CHAPITRE 13	
Les fichiers	541
1. Généralités concernant le traitement des fichiers	542
1.1 Notion d'enregistrement	542
1.2 Archivage de l'information sous forme binaire ou formatée	542
1.3 Accès séquentiel ou accès direct	543
1.4 Fichiers et implémentation	544
2. Le traitement des fichiers en C	544
2.1 L'absence de la notion d'enregistrement en C.	545
2.2 Notion de flux.	545
2.3 Distinction entre fichier binaire et fichier formaté.	547

2.4 Opérations applicables à un fichier et choix du mode d'ouverture.	549
2.5 Accès séquentiel et accès direct	551
2.6 Le tampon et sa gestion	551
3. Le traitement des erreurs de gestion de fichier	552
3.1 Introduction	552
3.2 La détection des erreurs en C	553
4. Les entrées-sorties binaires : fwrite et fread	556
4.1 Exemple introductif de création séquentielle d'un fichier binaire	556
4.2 Exemple introductif de liste séquentielle d'un fichier binaire	559
4.3 La fonction fwrite	561
4.4 La fonction fread	564
5. Les opérations formatées avec fprintf, fscanf, fputs et fgets.	568
5.1 Exemple introductif de création séquentielle d'un fichier formaté.	569
5.2 Exemple introductif de liste séquentielle d'un fichier formaté.	571
5.3 La fonction fprintf	573
5.4 La fonction fscanf.	575
5.5 La fonction fputs.	579
5.6 La fonction fgets	582
6. Les opérations mixtes portant sur des caractères	585
6.1 La fonction fputc et la macro putc	586
6.2 La fonction fgetc et la macro getc	589
7. L'accès direct.	593
7.1 Exemple introductif d'accès direct à un fichier binaire existant.	593
7.2 La fonction fseek.	595
7.3 La fonction ftell	597
7.4 Les possibilités de l'accès direct.	598
7.5 Détection des erreurs supplémentaires liées à l'accès direct	600
7.6 Exemple d'accès indexé à un fichier formaté	603
7.7 Les fonctions fsetpos et fgetpos	605
8. La fonction fopen et les différents modes d'ouverture d'un fichier	605
8.1 Généralités	605
8.2 La fonction fopen	606
9. Les flux prédéfinis.	609

CHAPITRE 14

La gestion dynamique	611
1. Intérêt de la gestion dynamique	611
2. Exemples introductifs	612
2.1 Allocation et utilisation d'un objet de type double	612
2.2 Cas particulier d'un tableau	614
3. Caractéristiques générales de la gestion dynamique	615
3.1 Absence de typage des objets	616
3.2 Notation des objets	616
3.3 Risques et limitations	617
3.4 Limitations	618
4. La fonction malloc	618
4.1 Prototype	618
4.2 La valeur de retour et la gestion des erreurs	619
5. La fonction free	620
6. La fonction calloc	620
6.1 Prototype	621
6.2 Rôle	621
6.3 Valeur de retour et gestion des erreurs	621
6.4 Précautions	622
7. La fonction realloc	623
7.1 Exemples introductifs	623
7.2 Prototype	624
7.3 Rôle	624
7.4 Valeur de retour	625
7.5 Précautions	626
8. Techniques utilisant la gestion dynamique	626
8.1 Gestion de tableaux dont la taille n'est connue qu'au moment de l'exécution	626
8.2 Gestion de tableaux dont la taille varie pendant l'exécution	627
8.3 Gestion de listes chaînées	628

CHAPITRE 15

Le préprocesseur	631
1. Généralités.	631
1.1 Les directives tiennent compte de la notion de ligne	631
1.2 Les directives et le caractère #	632
1.3 La notion de token pour le préprocesseur.	632
1.4 Classification des différentes directives du préprocesseur	633
2. La directive de définition de symboles et de macros	634
2.1 Exemples introductifs.	634
2.2 La syntaxe de la directive #define	637
2.3 Règles d'expansion d'un symbole ou d'une macro	640
2.4 L'opérateur de conversion en chaîne : #	650
2.5 L'opérateur de concaténation de tokens : ##.	653
2.6 Exemple faisant intervenir les deux opérateurs # et ##	655
2.7 La directive #undef.	655
2.8 Précautions à prendre	656
2.9 Les symboles prédéfinis	660
3. Les directives de compilation conditionnelle	661
3.1 Compilation conditionnelle fondée sur l'existence de symboles	661
3.2 Compilation conditionnelle fondée sur des expressions	664
3.3 Imbrication des directives de compilation conditionnelle	670
3.4 Exemples d'utilisation des directives de compilation conditionnelle	670
4. La directive d'inclusion de fichier source	672
4.1 Généralités	672
4.2 Syntaxe	673
4.3 Précautions à prendre	674
5. Directives diverses	677
5.1 La directive vide	677
5.2 La directive #line	678
5.3 La directive #error	678
5.4 La directive #pragma	679

CHAPITRE 16

Les déclarations	681
1. Généralités	681
1.1 Les principaux éléments : déclarateur et spécificateur de type	682
1.2 Les autres éléments	683
2. Syntaxe générale d'une déclaration	684
2.1 Forme générale d'une déclaration	685
2.2 Spécificateur de type structure	686
2.3 Spécificateur de type union	688
2.4 Spécificateur de type énumération	689
2.5 Déclarateur	689
3. Définition de fonction	691
3.1 Forme moderne de la définition d'une fonction	691
3.2 Forme ancienne de la définition d'une fonction	692
4. Interprétation de déclarations	692
4.1 Les règles	692
4.2 Exemples	693
5. Écriture de déclarateurs	695
5.1 Les règles	695
5.2 Exemples	696

CHAPITRE 17

Fiabilisation des lectures au clavier	699
1. Généralités	699
2. Utilisation de scanf	701
3. Utilisation de gets	702
4. Utilisation de fgets	703
4.1 Pour éviter le risque de débordement en mémoire	703
4.2 Pour ignorer les caractères excédentaires	704
4.3 Pour traiter l'éventuelle fin de fichier et paramétrer la taille des chaînes lues	706

CHAPITRE 18

Les catégories de caractères

et les fonctions associées	709
1. Généralités	709
1.1 Dépendance de l'implémentation et de la localisation	709
1.2 Les fonctions de test	710
2. Les catégories de caractères	710
3. Exemples	713
3.1 Pour obtenir la liste de tous les caractères imprimables et leur code	713
3.2 Pour connaître les catégories des caractères d'une implémentation	714
4. Les fonctions de transformation de caractères	715

CHAPITRE 19

Gestion des gros programmes	717
1. Utilisation de variables globales	718
1.1 Avantages des variables globales	719
1.2 Inconvénients des variables globales	719
1.3 Conseils en forme de compromis	720
2. Partage d'identificateurs entre plusieurs fichiers source	721
2.1 Cas des identificateurs de fonctions	721
2.2 Cas des identificateurs de types ou de synonymes	722
2.3 Cas des variables globales	723

CHAPITRE 20

Les arguments variables	725
1. Écriture de fonctions à arguments variables	725
1.1 Exemple introductif	725
1.2 Arguments variables, forme d'en-tête et déclaration	727
1.3 Contraintes imposées par la norme	728
1.4 Syntaxe et rôle des macros <code>va_start</code> , <code>va_arg</code> et <code>va_end</code>	729
2. Transmission d'une liste variable	730
3. Les fonctions <code>vprintf</code>, <code>vfprintf</code> et <code>vsprintf</code>	732

CHAPITRE 21

Communication avec l'environnement	735
1. Cas particulier des programmes autonomes	735
2. Les arguments reçus par la fonction main	736
2.1 L'en-tête de la fonction main	736
2.2 Récupération des arguments reçus par la fonction main	737
3. Terminaison d'un programme	738
3.1 Les fonctions exit et atexit	738
3.2 L'instruction return dans la fonction main	739
4. Communication avec l'environnement	740
4.1 La fonction getenv	740
4.2 La fonction system	741
5. Les signaux	741
5.1 Généralités	741
5.2 Exemple introductif	742
5.3 La fonction signal	744
5.4 La fonction raise	746

CHAPITRE 22

Les caractères étendus	747
1. Le type wchar_t et les caractères multioctets	748
2. Notation des constantes du type wchar_t	749
3. Les fonctions liées aux caractères étendus mblen, mbtowc et wctomb ...	750
3.1 Généralités	750
3.2 La fonction mblen	750
3.3 La fonction mbtowc	751
3.4 La fonction wctomb	751
4. Les chaînes de caractères étendus	752
5. Représentation des constantes chaînes de caractères étendus	753
6. Les fonctions liées aux chaînes de caractères étendus : mbstowcs et wctombs	753
6.1 La fonction mbstowcs	754
6.2 La fonction wctombs	754

CHAPITRE 23

Les adaptations locales	755
1. Le mécanisme de localisation	755
2. La fonction setlocale	756
3. La fonction localeconv	758

CHAPITRE 24

La récursivité	761
1. Notion de récursivité	761
2. Exemple de fonction récursive	762
3. L'empilement des appels	763
4. Autre exemple de récursivité	765

CHAPITRE 25

Les branchements non locaux	769
1. Exemple introductif	769
2. La macro setjmp et la fonction longjmp	770
2.1 Prototypes et rôles	770
2.2 Contraintes d'utilisation	771

CHAPITRE 26

Les incompatibilités entre C et C++	773
1. Les incompatibilités raisonnables	773
1.1 Définition d'une fonction	773
1.2 Les prototypes en C++	774
1.3 Fonctions sans valeur de retour	774
1.4 Compatibilité entre le type void * et les autres pointeurs	774
1.5 Les déclarations multiples	775
1.6 L'instruction goto	775
1.7 Initialisation de tableaux de caractères	776
2. Les incompatibilités incontournables	776
2.1 Fonctions sans arguments	776
2.2 Le qualifieur const	776
2.3 Les constantes de type caractère	778

ANNEXE A

La bibliothèque standard C90	779
1. Généralités	780
1.1 Les différents fichiers en-tête	780
1.2 Redéfinition d'une macro standard par une fonction	781
2. Assert.h : macro de mise au point	781
3. Ctype.h : tests de caractères et conversions majuscules - minuscules	782
3.1 Les fonctions de test d'appartenance d'un caractère à une catégorie	782
3.2 Les fonctions de transformation de caractères	783
4. Errno.h : gestion des erreurs	783
4.1 Constantes prédéfinies	783
4.2 Macros	783
5. Locale.h : caractéristiques locales	784
5.1 Types prédéfinis	784
5.2 Constantes prédéfinies	784
5.3 Fonctions	784
6. Math.h : fonctions mathématiques	784
6.1 Constantes prédéfinies	784
6.2 Traitement des conditions d'erreur	785
6.3 Fonctions trigonométriques	785
6.4 Fonctions hyperboliques	786
6.5 Fonctions exponentielle et logarithme	786
6.6 Fonctions puissance	787
6.7 Autres fonctions	787
7. Setjmp.h : branchements non locaux	788
7.1 Types prédéfinis	788
7.2 Fonctions et macros	788
8. Signal.h : traitement de signaux	788
8.1 Types prédéfinis	788
8.2 Constantes prédéfinies	788
8.3 Fonctions de traitement de signaux	789
9. Stdarg.h : gestion d'arguments variables	789
9.1 Types prédéfinis	789
9.2 Macros	789

10. Stddef.h : définitions communes	790
10.1 Types prédéfinis	790
10.2 Constantes prédéfinies	790
10.3 Macros prédéfinies	790
11. Stdio.h : entrées-sorties	790
11.1 Types prédéfinis	790
11.2 Constantes prédéfinies	790
11.3 Fonctions d'opérations sur les fichiers	791
11.4 Fonctions d'accès aux fichiers	792
11.5 Fonctions d'écriture formatée	793
11.6 Fonctions de lecture formatée	794
11.7 Fonctions d'entrées-sorties de caractères	795
11.8 Fonctions d'entrées-sorties sans formatage	797
11.9 Fonctions agissant sur le pointeur de fichier	798
11.10 Fonctions de gestion des erreurs d'entrée-sortie	799
12. Stdlib.h : utilitaires	800
12.1 Types prédéfinis	800
12.2 Constantes prédéfinies	800
12.3 Fonctions de conversion de chaîne	800
12.4 Fonctions de génération de séquences de nombres pseudo aléatoires	802
12.5 Fonctions de gestion de la mémoire	803
12.6 Fonctions de communication avec l'environnement	804
12.7 Fonctions de tri et de recherche	805
12.8 Fonctions liées à l'arithmétique entière	805
12.9 Fonctions liées aux caractères étendus	806
12.10 Fonctions liées aux chaînes de caractères étendus	806
13. String.h : manipulations de suites de caractères	807
13.1 Types prédéfinis	807
13.2 Constantes prédéfinies	807
13.3 Fonctions de copie	807
13.4 Fonctions de concaténation	808
13.5 Fonctions de comparaison	808
13.6 Fonctions de recherche	809
13.7 Fonctions diverses	810

14. Time.h : gestion de l'heure et de la date	811
14.1 Types prédéfinis	811
14.2 Constantes prédéfinies	812
14.3 Fonctions de manipulation de temps	812
14.4 Fonctions de conversion	813

ANNEXE B

Les normes C99 et C11	815
1. Contraintes supplémentaires (C99)	815
1.1 Type de retour d'une fonction	815
1.2 Déclaration implicite d'une fonction	815
1.3 Instruction return	816
2. Division d'entiers (C99)	816
3. Emplacement des déclarations (C99)	816
4. Commentaires de fin de ligne (C99)	817
5. Tableaux de dimension variable (C99, facultatif en C11)	818
5.1 Dans les déclarations	818
5.2 Dans les en-têtes de fonctions et leurs prototypes	818
6. Nouveaux types (C99)	819
6.1 Nouveau type entier long long (C99)	819
6.2 Types entiers étendus (C99)	819
6.3 Nouveaux types flottants (C99)	820
6.4 Le type booléen (C99)	820
6.5 Les types complexes (C99, facultatif en C11)	821
7. Nouvelles fonctions mathématiques (C99)	822
7.1 Généralisation aux trois types flottants (C99)	823
7.2 Nouvelles fonctions (C99)	823
7.3 Fonctions mathématiques génériques (C99)	824
8. Les fonctions en ligne (C99)	825
9. Les caractères étendus (C99) et Unicode (C11)	826
10. Les pointeurs restreints (C99)	826
11. La directive #pragma (C99)	827

12. Les calculs flottants (C99)	827
12.1 La norme IEEE 754	827
12.2 Choix du mode d'arrondi	828
12.3 Gestion des situations d'exception	828
12.4 Manipulation de l'ensemble de l'environnement de calcul flottant.	829
13. Structures incomplètes (C99)	829
14. Structures anonymes (C11)	829
15. Expressions fonctionnelles génériques (C11)	830
16. Gestion des contraintes d'alignement (C11)	830
17. Fonctions vérifiant le débordement mémoire (C11 facultatif)	830
18. Les threads (C11 facultatif)	831
19. Autres extensions de C99	831
20. Autres extensions de C11	831
Index	833