

INFORMATIQUE
TECHNIQUE



3^{ème} édition

Expert

C en action

Téléchargement
www.editions-eni.fr



Yves METTIER

eni
Editions

Les éléments à télécharger sont disponibles à l'adresse suivante :
<http://www.editions-eni.fr>
Saisissez la référence de l'ouvrage **EI3CACT** dans la zone de recherche
et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

Introduction

1. Avant-propos	15
2. Public visé	16
3. Organisation de ce livre	17
4. Notes de l'auteur	19
4.1 De la langue française	19
4.2 À propos du mot espace	20
4.3 De la langue utilisée dans les exemples	20
5. Conventions utilisées	21
6. Les exemples	22
7. Remerciements	22

Chapitre 1 Bases du langage C

1. Introduction	25
2. Écrire son premier programme	25
3. Commenter du code	28
4. Utiliser des variables et des chaînes de caractères	30
5. Définir et utiliser des constantes	34
6. Utiliser les opérations de base	37
7. Utiliser les opérateurs mathématiques	41
8. Afficher une expression	42

9. Exécuter du code de façon conditionnelle	48
10. Exécuter du code en boucle	52
11. Définir et utiliser un tableau	56
12. Définir et utiliser une structure	60
13. Définir et utiliser un pointeur	64
14. Définir et utiliser une union	68
15. Définir et utiliser une fonction	69
16. Passer des paramètres à une fonction	74
17. Utiliser une fonction de façon récursive	75
18. Écrire un fichier d'en-têtes	77
19. Compiler une section de code en fonction d'une condition.	81
20. Exécuter une fonction avant la fin du programme	83

Chapitre 2 **Outils de compilation**

1. Introduction	87
2. Compiler un programme	87
3. Automatiser la compilation avec make.	90
4. Bien répartir le code sur plusieurs fichiers.	92
5. Obtenir des binaires optimisés	94
6. Passer des paramètres au préprocesseur	97

Chapitre 3 **Bibliothèques et fonctions**

1. Introduction	99
2. Créer et utiliser une bibliothèque avec les outils GNU	100
3. Charger une bibliothèque de manière dynamique	103

- 4. Lancer une fonction d'une bibliothèque dynamique 105
- 5. Écrire un greffon 106
- 6. Lancer une fonction du programme depuis la bibliothèque dynamique 115
- 7. Créer une bibliothèque dynamique liée statiquement aux bibliothèques dont elle dépend 116

Chapitre 4
Débogage d'un programme

- 1. Introduction 119
- 2. Déboguer proprement avec printf() 120
- 3. Déterminer où le programme stoppe 122
- 4. Afficher le contenu d'une variable 128
- 5. Retarder l'arrêt intempestif d'un programme pour sauvegarder des données 131

Chapitre 5
L'art de programmer en C

- 1. Limiter les risques d'erreurs de segmentation 137
- 2. Écrire du code réutilisable 142
- 3. Faire renvoyer plusieurs valeurs à une fonction 148
- 4. Le bon usage des macros 151
- 5. Goto et les traitements d'exception 155

Chapitre 6 **Gestion des erreurs**

1. Introduction	159
1.1 Fautes de frappe	159
1.2 Les erreurs sur la mémoire	161
1.3 Fichiers inexistant, impossibles à créer et autres problèmes liés aux fichiers	162
1.4 Résumé des comportements possibles face à une erreur	163
2. Récupérer le code d'erreur	163
3. Récupérer le descriptif de l'erreur	165
4. Garder une trace de l'exécution d'un programme	166
5. Créer une fonction de journalisation	169
6. Utiliser syslog	173

Chapitre 7 **Structures de données**

1. Introduction	177
2. Choisir une structure pour une liste de données	178
3. Choisir une structure pour des données sous forme clé-valeur	188
4. Choisir une structure pour des données sous forme de graphe	191
5. Coder un tableau dont l'index est une chaîne de caractères	194
6. Choisir une structure pour un tableau dont l'index est un entier	196
7. Optimiser la recherche d'une aiguille dans une botte de foin	198
8. Trier une liste selon un critère donné	200
9. Supprimer les doublons dans une structure de données	203

Chapitre 8
Dates et heures

- 1. Introduction207
- 2. Récupérer la date et l'heure courante208
- 3. Connaître le jour de la semaine210
- 4. Effectuer des calculs sur les dates211
- 5. Convertir une estampille en struct tm et réciproquement214
- 6. Convertir une estampille ou un struct tm en chaîne de caractères.215
- 7. Convertir une chaîne de caractères en estampille ou en struct tm .216
- 8. Faire une pause221
- 9. Calculer le temps mis par un extrait de programme à s'exécuter ..224

Chapitre 9
Chaînes de caractères

- 1. Introduction229
- 2. Allouer la mémoire pour une chaîne de caractères230
- 3. Copier une chaîne de caractères233
- 4. Analyser une chaîne de caractères, caractère par caractère235
- 5. Concaténer deux chaînes de caractères235
- 6. Convertir un nombre en chaîne de caractères et réciproquement. .242
- 7. Transformer une chaîne de caractères avec des retours chariot
en un tableau de chaînes de caractères247
- 8. Découper une chaîne de caractères en fonction d'un séparateur. .251
- 9. Récupérer le chemin et le nom d'un fichier
spécifiés dans une chaîne de caractères253
- 10. Remplacer une sous-chaîne par une autre
sous-chaîne dans une chaîne de caractères255

11. Déterminer si une chaîne de caractères est contenue dans une autre et à quelle position	257
12. Déterminer si une chaîne de caractères correspond au motif précisé dans une expression régulière	260
13. Trouver le nombre d'occurrences d'une chaîne de caractères dans une autre	265
14. Traiter les blancs au début et à la fin d'une chaîne	266
15. Transformer tout séparateur par une espace	268
16. Transformer une chaîne en minuscules ou en majuscules	269

Chapitre 10 **Gestion de la mémoire**

1. Introduction	273
2. Allouer de la mémoire	275
3. Créer son gestionnaire de mémoire	279
4. Redéfinir les fonctions d'allocation de mémoire	283
5. Tracer les allocations de mémoire	284
6. Créer de la mémoire partagée entre processus	292

Chapitre 11 **Gestion des répertoires et des fichiers**

1. Introduction	297
2. Connaître le contenu d'un répertoire	297
3. Effectuer une opération récursivement sur tous les fichiers d'un répertoire et de ses sous-répertoires	299
4. Effacer un répertoire et tout son contenu	305
5. Obtenir des informations sur un fichier	306
6. Modifier la date de dernière modification d'un fichier	309

7. Créer un fichier 310
8. Renommer un fichier 311
9. Copier un fichier 312
10. Déplacer un fichier 314
11. Supprimer un fichier 315
12. Créer un répertoire 316
13. Créer un lien symbolique 318
14. Obtenir le répertoire courant 319
15. Reconnaître que deux noms correspondent au même fichier 321

Chapitre 12

Contenu des fichiers

1. Introduction 323
2. Lire un fichier 324
3. Écrire dans un fichier 330
4. Lire un fichier de configuration simple 332
5. Rechercher une donnée dans un fichier texte 338
6. Ajouter des données à un fichier 340
7. Remplacer des données dans un fichier 345
8. Supprimer une partie d'un fichier 346
9. Calculer combien de lignes ou de caractères contient un fichier . . . 348
10. Calculer la taille d'un fichier 349
11. Effectuer une lecture non bloquante d'un fichier 350
12. Classer un fichier texte 351
13. Lire un fichier au format DOS 355
14. Poser un verrou sur un fichier 355
15. Créer des fichiers temporaires 358

16. Lire en continu dans un fichier qui croît	359
---	-----

Chapitre 13

Réseau

1. Introduction	361
1.1 Les couches réseau	361
1.2 Les protocoles IP, TCP et UDP	363
1.3 Les sockets.	364
1.4 Choisir entre TCP/IP et UDP/IP	365
1.5 Remarque sur TCP/IP	366
1.6 Au programme	366
2. Créer un serveur TCP/IP	366
3. Créer un client TCP/IP.	371
4. Créer un client et un serveur UDP/IP	375
5. Sécuriser une connexion avec SSL/TLS.	381
6. Connaître le nom et l'adresse IP de ma machine	393
7. Connaître l'adresse IP d'une machine à partir de son nom	394
8. Créer un serveur TCP/IP multi-processus.	396
9. Créer un serveur TCP/IP multi-thread	400
10. Créer un serveur TCP/IP mono-processus sans thread	408
11. Créer un serveur TCP/IP ou UDP/IP qui utilise le démon inetd	414
12. Résoudre le problème des architectures petit et gros boutistes	418
13. Modifier les options sur une socket.	419

Chapitre 14
Protocoles réseau

- 1. Introduction 421
- 2. Faire suivre un port. 425
- 3. Obtenir une page web d'un serveur HTTP ou HTTPS 429
- 4. Télécharger et transférer des fichiers avec le protocole FTP 438
- 5. Créer un serveur HTTP 443
- 6. Envoyer un courrier électronique 453
- 7. Récupérer un message électronique sur un serveur POP3 460
- 8. Récupérer un message électronique sur un serveur IMAP. 473
- 9. Rechercher et lire un courrier dans une boîte au format maildir. . . 488
- 10. Rechercher et lire un courrier dans une boîte au format mbox. . . . 496

Chapitre 15
Signaux

- 1. Introduction 501
- 2. Mettre en place un gestionnaire de signaux 503
- 3. Bloquer des signaux 510
- 4. Connaître le masque de blocage des signaux 512
- 5. Savoir si un signal a été bloqué 512
- 6. Intercepter [Ctrl] C 513
- 7. Envoyer un signal 514
- 8. Sauvegarder un gestionnaire de signaux
et le restaurer par la suite. 515
- 9. Limiter le temps d'exécution d'une partie d'un programme 516

Chapitre 16**Exécution parallèle**

1. Introduction 519
2. Créer un nouveau processus 521
3. Éviter les processus zombies 523
4. Créer un nouveau thread 525
5. Limiter l'accès à une section critique 531
6. Communiquer entre deux processus distincts 538

Chapitre 17**Système et processus**

1. Introduction 543
2. Lancer un programme 544
3. Lancer un script Perl 547
4. Récupérer le code de retour d'un programme qui s'est terminé. 556
5. Récupérer la sortie standard d'un programme 558
6. Récupérer les arguments passés sur la ligne de commande 561
7. Envoyer des données sur l'entrée standard d'un programme. 574
8. Lire une chaîne de caractères depuis l'entrée standard 576
9. Lire un mot de passe sur l'entrée standard 578
10. Partager un identifiant de fichier entre deux programmes 579
11. Connaître le PID du processus et celui de son père 583
12. Créer un démon 584
13. Connaître l'utilisateur qui a lancé le programme 587
14. Changer d'identité 587

Chapitre 18
Internationalisation

- 1. Introduction591
- 2. Internationaliser un programme avec gettext592
- 3. Traduire un programme internationalisé avec gettext596
- 4. Maintenir à jour la liste des chaînes à traduire.....599
- 5. Éviter certains pièges linguistiques600

Chapitre 19
Compression

- 1. Introduction605
- 2. Lire un fichier compressé607
- 3. Écrire un fichier compressé612
- 4. Compresser des données en mémoire617
- 5. Décompresser des données en mémoire619
- 6. Décompresser un fichier tar.gz ou tar.bz2622

Chapitre 20
XML avec libxml2

- 1. Introduction629
- 2. Lire un document XML632
- 3. Transformer un arbre DOM en XML636
- 4. Ajouter un nœud à un arbre DOM637
- 5. Modifier un nœud d'un arbre DOM639
- 6. Supprimer un nœud d'un arbre DOM.....641
- 7. Parcourir un arbre DOM642

8. Rechercher un nœud ou un ensemble de nœuds avec XPath dans un arbre DOM	643
9. Créer et utiliser un fichier de configuration en XML.....	647
10. Parcourir un document XML avec SAX	650
11. Rechercher une donnée dans un document XML (SAX).....	656
12. Lire un flux de données XML.....	658
13. Transformer un document XML avec XSLT	660

Chapitre 21

Bases de données

1. Introduction	665
2. Effectuer une requête sur un serveur PostgreSQL	666
3. Effectuer une requête sur un serveur MySQL	680
4. Effectuer une requête sur une base de données SQLite3	694
5. Effectuer une requête sur un serveur de bases de données compatible ODBC	706

Chapitre 22

Automatisation de la compilation

1. Introduction	723
2. Démarrer un projet avec autoconf et automake.....	725
3. Ajouter des tests de fonctions ou de bibliothèques	729
4. Récupérer les variables de la commande ./configure	733
5. Passer des options supplémentaires au compilateur	736
6. Prendre en compte l'internationalisation d'un projet.....	738
7. Utiliser autoconf, automake et libtool pour créer une bibliothèque	742

8. Créer une bibliothèque et l'utiliser dans un projet
avec autoconf et automake745

Chapitre 23

C11 et les normes du langage C

1. Introduction747
2. Obtenir la norme C11749
3. Programmer et compiler avec la norme C11750
4. Principaux ajouts et modifications apportés par la norme C11 ...753

Index761

Chapitre 11

Gestion des répertoires et des fichiers

1. Introduction

Sur les systèmes Unix, il est souvent dit que tout est fichier. Cela s'applique notamment aux répertoires. Et les fonctions utilisables pour les fichiers le sont aussi pour les répertoires. Cependant, les répertoires étant des fichiers particuliers puisque le contenu n'est rien d'autre que la liste des fichiers qu'ils contiennent, nous disposons de quelques fonctions supplémentaires qui nous permettent de les ouvrir, de lire leur contenu entrée par entrée, de les créer simplement... Ce chapitre traitera des répertoires, ainsi que de leur contenu, les fichiers. Vous trouverez donc le nécessaire pour agir sur ceux-ci, à l'exception du contenu des fichiers qui fait l'objet du chapitre suivant.

2. Connaître le contenu d'un répertoire

Problème

Vous voulez connaître les noms des fichiers et des répertoires contenus dans un répertoire.

Solution

Servez-vous des fonctions `opendir()`, `readdir()` et `closedir()`. Utilisez-les de manière récursive dans les sous-répertoires si vous voulez aussi les parcourir.

Discussion

L'utilisation d'`opendir()` est simple. La difficulté réside dans ce que nous souhaitons faire des noms des fichiers et des répertoires obtenus avec `readdir()`. Voici un cas d'école, un exemple qui affiche les noms des fichiers et sous-répertoires d'un répertoire :

```
int
printdir (char *dirname)
{
    DIR *FD;
    struct dirent *f;

    if (NULL == (FD = opendir (dirname)))
    {
        fprintf (stderr, "opendir() impossible\n");
        return (-1);
    }
    printf ("%s :\n", dirname);
    while ((f = readdir (FD))
    {
        printf ("  %s\n", f->d_name);
    }
    closedir (FD);
    return (0);
}
```

Dans la boucle qui affiche les fichiers et répertoires, recourrez à la recette "Obtenir des informations sur un fichier" pour afficher plus que le nom des fichiers, en particulier s'il s'agit d'un fichier ou d'un répertoire. Cela permet, pour parcourir le répertoire de manière récursive, de réitérer le processus sur les répertoires ainsi rencontrés. La recette suivante applique cette méthode pour effectuer une opération récursivement sur tous les fichiers d'un répertoire et de ses sous-répertoires. S'il s'agit d'afficher le contenu d'un répertoire et de ses sous-répertoires, faites appel à cette recette, l'action étant d'afficher le nom du fichier.

Prototypes

```
#include <sys/types.h>
#include <dirent.h>

DIR *opendir (const char *filename);

struct dirent *readdir (DIR * dirp);

int closedir (DIR * dirp);
```

■ Remarque

Voir aussi les pages de manuel de `opendir()`, `readdir()` et `closedir()`.

3. Effectuer une opération récursivement sur tous les fichiers d'un répertoire et de ses sous-répertoires

Problème

Vous souhaitez effectuer une opération sur tous les fichiers d'un répertoire ainsi que ses sous-répertoires.

Solution

Il n'y a pas de solution toute faite, vous devez programmer un algorithme de parcours des répertoires de manière récursive.

Discussion

Il existe plusieurs manières de parcourir les répertoires. La méthode la plus simple consiste à tester chaque fichier pour savoir si c'est un répertoire, et le cas échéant, appliquer l'algorithme à ce répertoire, de manière récursive. Cette méthode a l'avantage d'être simple mais pose un problème d'ordre de parcours. En effet, souhaitez-vous parcourir tous les fichiers et répertoires d'un même niveau ou pouvez-vous vous contenter de traiter les sous-répertoires d'un niveau au moment où ils se présentent, et ne continuer le niveau en cours que plus tard ?

Le programme suivant montre une méthode proche de celle de la commande `find` sur GNU/Linux, à base de récursivité et en utilisant `chdir()` qui nous place en permanence dans le répertoire en cours de traitement. Cette méthode utilise une liste chaînée pour obtenir la liste des fichiers d'un répertoire. Cela permet de faire un classement optionnel avant traitement. Dans notre cas, le classement consiste à traiter d'abord les fichiers, puis à aller dans les sous-répertoires de manière récursive.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>
#include <errno.h>

/*****
 * Action sur les fichiers et répertoires *
 *****/

/* Effectue une action sur un répertoire. */
void
action_dir (const char *dir)
{
    printf ("%s\n", dir);
}

/* Effectue une action sur un répertoire avant parcours récursif
 * de son contenu.
 */
void
action_dir_pre (const char *root, const char *dir)
{
    printf ("\n-> %s/%s\n", root, dir);
}

/* Effectue une action sur un répertoire après parcours récursif
 * de ce répertoire.
 */
void
action_dir_post (const char *root, const char *dir)
{
}
```

```
/* Effectue une action sur un fichier. */
void
action_file (const char *file)
{
    printf ("%s\n", file);
}

/*****
 * Définition d'une liste chaînée simple *
 *****/

typedef struct slist_t
{
    char *name;
    int is_dir;
    struct slist_t *next;
} slist_t;

/*****
 * Parcours récursif des répertoires *
 *****/

int
recursive_dir (char *root, char *dirname)
{
    slist_t *names = NULL;
    slist_t *sl;

    DIR *FD;
    struct dirent *f;
    int cwrlen = 32;
    char *cwd;
    char *new_root;

    if (NULL == (cwd = malloc (cwrlen * sizeof *cwd)))
    {
        fprintf (stderr, "Problème avec malloc\n");
        exit (EXIT_FAILURE);
    }

    /* Concaténation new_root = "root/dirname" */
    if (root)
    {
        int rootlen = strlen (root);
        int dirnamelen = strlen (dirname);
        if (NULL ==
```

```

        (new_root =
         malloc ((rootlen + dirnamelen + 2) * sizeof *new_root))
    {
        fprintf (stderr, "Problème avec malloc\n");
        exit (EXIT_FAILURE);
    }
    memcpy (new_root, root, rootlen);
    new_root[rootlen] = '/';
    memcpy (new_root + rootlen + 1, dirname, dirnamelen);
    new_root[rootlen + dirnamelen + 1] = '\0';
}
else
    new_root = strdup (dirname);

/* Obtention du répertoire courant */
while (NULL == (cwd = getcwd (cwd, cwdlen)))
{
    if (ERANGE != errno)
    {
        fprintf (stderr, "Problème avec getcwd (errno = '%s')\n",
                strerror (errno));
        exit (EXIT_FAILURE);
    }
    cwdlen += 32;
    cwd = realloc (cwd, cwdlen * sizeof *cwd);
}
chdir (dirname);

/* Remplissage de la liste chaînée avec les noms des fichiers
 * du répertoire courant. */
if (NULL == (FD = opendir (".")))
{
    fprintf (stderr, "opendir() impossible\n");
    return (-1);
}
sl = names;
while ((f = readdir (FD))
{
    struct stat st;
    slist_t *n;
    if (!strcmp (f->d_name, "."))
        continue;
    if (!strcmp (f->d_name, ".."))
        continue;
    if (stat (f->d_name, &st))
        continue;

```