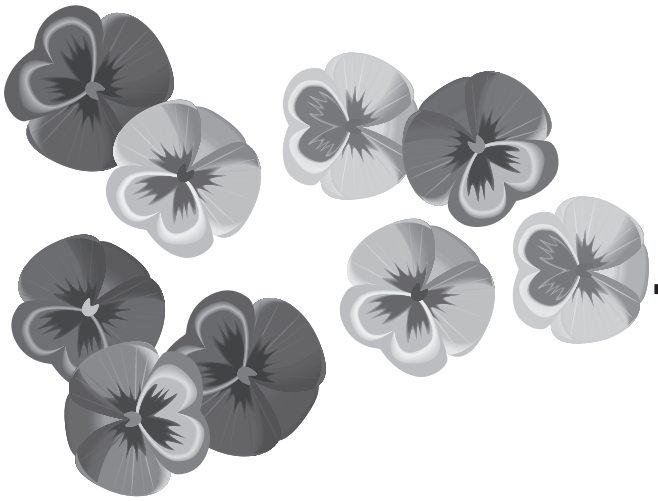

Tour d'horizon de la Partie I

Avant de commencer votre apprentissage du langage C, un compilateur et un éditeur sont nécessaires. Si vous n'avez ni l'un ni l'autre, vous pouvez quand même utiliser ce livre mais la valeur de son enseignement en sera diminuée. La meilleure façon d'apprendre un langage de programmation est de créer et lancer de nombreux programmes. Les exemples donnés dans ce livre offrent un bon support pour les définitions et exercices.

Chaque chapitre se termine par un atelier constitué d'un quiz et de quelques exercices portant sur les sujets étudiés. Les réponses et solutions complètes des premiers chapitres se trouvent dans l'Annexe I. Il n'a pas été possible de prévoir toutes les réponses pour les derniers chapitres car il existe un grand nombre de solutions. Nous vous recommandons de tirer le meilleur parti de ces ateliers et de contrôler vos réponses.

Ce que vous allez apprendre

Cette première partie aborde les notions de base du C. Les Chapitres 1 et 2 vous apprendront à créer un programme C et à en reconnaître les éléments de base. Le Chapitre 3 définit les différents types de variables C. Le Chapitre 4 introduit les instructions et expressions d'un programme pour obtenir de nouvelles valeurs. Il vous explique également comment introduire des conditions dans l'exécution d'un programme avec l'ordre IF. Le Chapitre 5 traite des fonctions du langage C et de la programmation structurée. Le Chapitre 6 concerne les commandes qui permettent de contrôler le déroulement des programmes. Enfin, le Chapitre 7 vous permettra d'imprimer et de dialoguer avec votre clavier ou votre écran.

Ce livre s'appuie sur le standard C ANSI(norme C89). La suppression de la fonction fgets() dans la norme C11 est également prise en compte. Cela signifie que vous pouvez utiliser le compilateur C de votre choix s'il respecte bien la norme ANSI, mais aussi un compilateur compatible C11.

1

Comment démarrer

Vous apprendrez dans ce chapitre :

- Pourquoi le langage C représente le meilleur choix d'un langage de programmation
- Les étapes du cycle de développement d'un programme
- Comment écrire, compiler et lancer votre premier programme C
- Comment faire face aux messages d'erreurs générés par le compilateur et l'éditeur de liens

Bref historique du langage C

Le langage C a été créé par Dennis Ritchie aux Bell Telephone Laboratories en 1972. Il a été conçu dans un dessein bien précis : développer le système d'exploitation UNIX, déjà utilisé sur de nombreux ordinateurs. Dès l'origine, il devait donc permettre aux programmeurs de travailler de manière productive et efficace.

En raison de sa puissance et de sa souplesse, l'utilisation du C s'est rapidement répandue au-delà des laboratoires Bell. Les programmeurs ont commencé à l'utiliser pour écrire toutes sortes de programmes. Rapidement, des organisations diverses ont utilisé leurs propres versions du langage C, et de subtiles différences d'implémentation sont devenues un véritable casse-tête pour les programmeurs. En réponse à ce problème, l'American National Standards Institute (ANSI) a formé un comité en 1983 pour établir une définition standard du C, qui est devenu le C standard ANSI, connu également avec le nom de la norme correspondante C89. À quelques exceptions près, les compilateurs C d'aujourd'hui adhèrent à ce standard. Ils devraient également être compatibles avec la norme C11 plus récente.

Le nom du langage C vient de son prédécesseur qui était appelé B. Le langage B a été développé par Ken Thompson qui travaillait aussi aux laboratoires Bell.

Pourquoi utiliser le langage C ?

Il existe de nombreux langages de programmation de haut niveau comme le C, le Pascal, ou le Basic. Ils sont tous excellents et conviennent pour la plupart des tâches de programmation. Toutefois, les professionnels placent le langage C en tête de liste pour plusieurs raisons :

- Il est souple et puissant. Ce que vous pourrez accomplir avec ce langage n'est limité que par votre imagination. Vous n'aurez aucune contrainte. Le langage C est utilisé pour des projets aussi variés que des systèmes d'exploitation, des traitements de textes, des graphiques, des tableurs ou même des compilateurs pour d'autres langages.
- Lorsqu'une nouvelle architecture (nouveau processeur, nouveau système d'exploitation...) apparaît, le premier langage disponible est généralement le C car contrairement à d'autres, il est facile à porter. De plus, un compilateur C est souvent disponible sur les ordinateurs (à l'exception de Windows malheureusement), ce qui n'est pas le cas pour les autres langages.
- Avec la norme ANSI, le C est devenu un langage portable. Cela signifie qu'un programme C écrit pour un type d'ordinateur (un PC IBM, par exemple) peut être compilé pour tourner sur un autre système (comme un DEC VAX) avec très peu ou aucune modification. Les règles qui sont à respecter par les compilateurs sont décrites plus loin dans ce livre.
- Le langage C contient peu de mots. Une poignée d'expressions appelées mots clés servent de bases pour l'élaboration des fonctions. On pourrait penser, à tort, qu'un langage possédant plus de mots clés (quelquefois appelés mots réservés) pourrait être plus puissant. Lorsque vous programmerez avec ce langage, vous vous apercevrez que vous pouvez réaliser n'importe quelle tâche.
- Le langage C est modulaire. Son code peut (et devrait) être écrit sous forme de sous-programmes appelés fonctions. Ces fonctions peuvent être réutilisées pour d'autres applications ou programmes. Si vous passez des informations à ces fonctions, vous obtenez du code réutilisable.

Comme vous pouvez le constater, le choix du C en tant que premier langage de programmation est excellent. Vous avez certainement entendu parler de C++. Ce langage s'appuie sur une technique de programmation appelée programmation orientée objet.

C++ était initialement une version améliorée du C, à savoir un C disposant de fonctions supplémentaires pour la programmation orientée objet. Le C++ est aujourd'hui un langage à part entière. Si vous êtes amenés à étudier ce langage, ce que vous aurez appris du C vous aidera grandement.

Un autre langage, également basé sur C, a été l'objet d'une attention toute particulière. Il s'agit de Java. Si vous décidez de vous orienter vers la programmation Java, vous découvrirez rapidement qu'il existe de nombreuses similitudes entre ces deux langages.

Avant de programmer

Vous ne pouvez résoudre que les problèmes que vous aurez identifiés. Il sera alors possible de bâtir un plan pour les corriger. Lorsque vous aurez appliqué ce plan, vous devrez tester les résultats pour savoir si les problèmes ont bien été résolus. Cette logique s'applique à de nombreux domaines, la programmation en fait partie.

Voici les étapes à suivre pour créer un programme en langage C (ou dans n'importe quel autre langage) :

1. Définir les objectifs du programme.
2. Choisir les méthodes que vous voulez utiliser pour écrire ce programme.
3. Créer le programme.
4. Enfin, l'exécuter et observer les résultats.

Un exemple d'objectif (voir étape 1) serait d'écrire un traitement de texte ou un programme de base de données. Un objectif plus simple consiste à afficher votre nom sur l'écran. Si vous n'avez pas de fonction à réaliser, vous n'avez pas besoin d'un programme.

Pour la deuxième étape, vous devez définir vos besoins, la formule à utiliser, et établir un ordre de traitement des informations.

Par exemple, imaginez que quelqu'un vous demande d'écrire un programme pour calculer l'aire d'un cercle. L'étape 1 est réalisée puisque vous connaissez votre objectif : trouver la valeur de cette aire. L'étape 2 consiste à déterminer quelles sont les données à connaître pour le calcul. Si l'utilisateur du programme donne le rayon du cercle, la formule πr^2 vous donnera la réponse. Vous pouvez maintenant passer aux étapes 3 et 4 qui constituent le développement du programme.

Cycle de développement du programme

La première étape du développement d'un programme est la création du code source avec un éditeur. La deuxième étape consiste à compiler ce code pour obtenir un fichier objet. Dans la troisième, vous transformez le code compilé en fichier exécutable. Le lancement du programme dans la quatrième étape permet d'en vérifier les résultats.

Création du code source

Le code source est une série de commandes ou de déclarations qui indiquent à l'ordinateur les tâches que vous voulez lui faire exécuter. C'est la première étape du développement et le code source est créé à l'aide d'un éditeur. Voici un exemple d'instruction de code source C :

```
printf("Bonjour, vous !");
```

Cette instruction demande à l'ordinateur d'afficher le message `"bonjour, vous !"` à l'écran.

Utilisation de l'éditeur

La plupart des compilateurs sont livrés avec un éditeur intégré qui permet de créer le code source. Consultez votre manuel pour savoir si votre compilateur en fait partie.

La plupart des systèmes d'exploitation contiennent un programme qui peut être utilisé comme un éditeur. Si vous travaillez avec UNIX, vous pouvez utiliser vi ou vim, emacs ou un bloc-notes comme gedit ou kedit. Microsoft Windows vous offre le bloc-notes. Les logiciels de traitement de texte utilisent des codes spéciaux pour formater leurs documents. Ces codes ne peuvent pas être lus correctement par les autres programmes. L'*American Standard Code for Information Interchange* (ASCII) a défini un format de texte standard que n'importe quel programme, y compris le C, peut utiliser. Beaucoup de traitements de texte, comme LibreOffice.org, Abiword, Koffice et Microsoft Word, sont capables de sauvegarder des fichiers source en format ASCII (comme un fichier texte plutôt que comme un fichier document). Pour obtenir un fichier en format ASCII avec un traitement de texte, vous devez choisir l'option de sauvegarde ASCII ou texte.

Vous n'êtes pas obligé d'utiliser un de ces éditeurs. Il existe des programmes, que vous pouvez acheter, qui sont spécialement destinés à créer du code source. Citons également les logiciels libres (et gratuits) Ajuta et Kdevelop disponibles au moins sur GNU/Linux.

Pour trouver des éditeurs différents, vous pouvez consulter votre revendeur local, les catalogues de vente par correspondance ou encore les petites annonces des magazines de programmation. Sur votre distribution Linux, effectuez une recherche dans les packages disponibles.

Quand vous sauvegardez un fichier source, il faut lui donner un nom. Vous pouvez choisir n'importe quel nom ou extension, mais il existe une convention : le nom du programme doit représenter la fonction de ce programme et `.c` est reconnue comme l'extension appropriée.

Compilation du code source

Votre ordinateur ne peut pas comprendre le code source C. Il ne peut comprendre que des instructions binaires dans ce que l'on appelle du langage machine. Votre programme C doit être transformé en langage machine pour pouvoir être exécuté sur votre ordinateur. Cela représente la deuxième étape de développement du programme. Cette opération est réalisée par un compilateur qui transforme votre fichier code source en un fichier contenant les mêmes

instructions en langage machine. Ce fichier créé par le compilateur contient le code objet , et on l'appelle fichier objet.

Ce livre s'appuie sur le standard C ANSI. Cela signifie que vous pouvez utiliser le compilateur C de votre choix s'il respecte bien la norme ANSI.

Chaque compilateur possède sa propre commande pour créer du code objet. En général, il faut taper la commande de lancement du compilateur suivie du nom du fichier source. Voici quelques exemples de commandes destinées à compiler le fichier source `radius.c` en utilisant divers compilateurs DOS/Windows :

Compilateur	*Commande*
Gnu gcc	`gcc radius.c`
C Microsoft	`cl radius.c`
Turbo C de Borland	`tcc radius.c`
C Borland	`bcc radius.c`
Compilateurs C Unix	`cc radius.c`

La compilation sera simplifiée dans un environnement de développement graphique. Dans la plupart des cas, cette opération sera réalisée à partir du menu ou de l'icône correspondante. Une fois le code compilé, il suffira alors de sélectionner l'icône en cours ou la touche du menu adéquate pour exécuter le programme. Pour de plus amples renseignements vous vous référerez au manuel de votre compilateur.

Après cette opération, vous trouverez dans votre répertoire courant un nouveau fichier ayant le même nom que votre fichier source, mais avec l'extension .o ou `.obj`. Cette extension sera reconnue par l'éditeur de liens comme celle d'un fichier objet. Avec certains compilateurs dont gcc, sans option spécifique, le fichier généré s'appellera a.out.

Création du fichier exécutable

Une partie du langage C est constituée d'une *bibliothèque de fonctions* contenant du code objet (ce code a déjà été compilé) destiné à des *fonctions prédéfinies* . Ces fonctions sont fournies avec votre compilateur et `printf()` en est un exemple.

Ces fonctions réalisent des tâches très souvent réalisées comme afficher des informations à l'écran ou lire un fichier. Si votre programme les utilise, le fichier objet obtenu après compilation doit être complété par le code objet issu de la bibliothèque de fonctions. Cette dernière étape, appelée liaison, fournit le programme exécutable (*exécutable* signifie que ce programme peut être exécuté sur votre ordinateur).

La Figure 1.1 représente le schéma de la transformation du code source en programme exécutable.

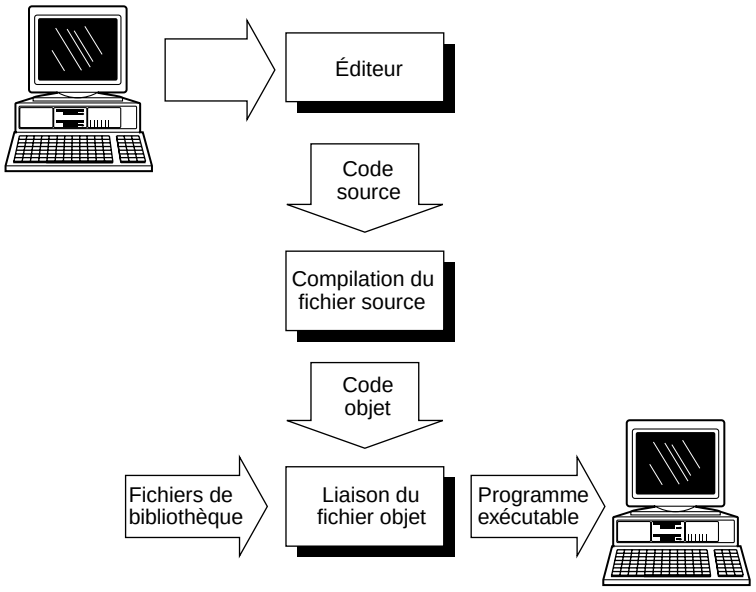

Figure 1.1
Le code source est transformé en code objet par le compilateur puis en fichier exécutable par l'éditeur de liens.

Fin du cycle de développement

Une fois que vous avez obtenu votre fichier exécutable, vous pouvez lancer votre programme en saisissant son nom à l'invite de votre système. Si les résultats obtenus sont différents de ceux recherchés, vous devez recommencer à la première étape. Il faut identifier l'origine du problème et corriger le code source. À chaque transformation de ce code, il est nécessaire de recompiler le programme et de relancer l'éditeur de liens (*linker* en anglais) pour créer une version corrigée du fichier exécutable. Répétez ces opérations jusqu'à ce que le programme s'exécute de façon correcte.

Bien que nous ayons différencié la compilation de la liaison, beaucoup de compilateurs exécutent ces deux opérations en une seule étape. Quelle que soit la méthode utilisée, ce sont bien deux actions séparées.

Cycle de développement

Étape 1	Utilisez un éditeur pour créer le code source. Par convention, ce fichier doit avoir l'extension `.c` (par exemple, monprog.c, database.c, etc.).
Étape 2	Compilez votre programme. Si le compilateur ne rencontre pas d'erreur dans votre code source, vous obtenez un fichier objet du même nom que votre fichier source avec une extension `.obj` ou `.o` (par exemple, monprog.c est compilé en monprog.o). Si le code source contient des erreurs, le compilateur échoue et vous les affiche pour correction.
Étape 3	Exécutez la liaison. Si aucune erreur n'apparaît, vous obtenez un programme exécutable dans un fichier du même nom que le fichier objet (avec une extension `.exe` sur Windows par exemple, monprog.obj devient monprog.exe).
Étape 4	Exécutez votre programme. Contrôlez les résultats obtenus et recommencez à l'étape 1 si des modifications sont nécessaires dans le fichier source.

Les étapes de développement du programme sont représentées dans la Figure 1.2. Il faut parcourir ce cycle jusqu'à obtenir le résultat recherché. Même le meilleur programmeur ne peut simplement s'asseoir et écrire un programme complet sans aucune erreur dès la première étape. C'est pourquoi il est important de maîtriser parfaitement ces outils : l'éditeur, le compilateur et l'éditeur de liens.

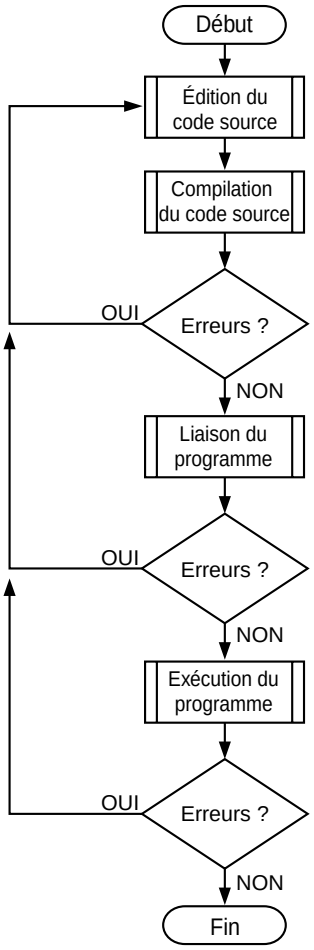

Figure 1.2
Les étapes de développement d'un programme C.

Votre premier programme C

Voici un exemple qui permettra de vous familiariser avec votre compilateur. Même si vous ne comprenez pas la syntaxe, cet exercice est là pour vous faire écrire, compiler, et exécuter un programme C.

Ce programme s'appelle `hello.c`, et il va afficher "Hello, world !" sur votre écran. Vous trouverez le code source de ce programme dans le Listing 1.1. Attention, vous ne devez pas ajouter les numéros de ligne ni les deux points qui suivent. Nous les avons ajoutés dans ce livre pour pouvoir donner la référence des lignes qui seront commentées.

Listing 1.1 : *hello.c*

```
1 : #include <stdio.h>
2 :
3 : int main()
4 : {
5 :    printf("Hello, World !\n");
6 :    return 0;
7 : }
```

Installez votre compilateur en suivant les instructions fournies avec le produit. Quel que soit votre système d'exploitation (Windows, Linux, etc.), assurez-vous d'avoir bien compris le fonctionnement du compilateur et de l'éditeur de votre choix. Vous pouvez maintenant suivre les étapes ci-après pour saisir, compiler, et exécuter hello.c.

Création et compilation de hello.c

Voici comment créer et compiler le programme hello.c :

1. Placez-vous sur le répertoire qui contient vos programmes C et démarrez votre éditeur. Comme nous l'avons mentionné précédemment, vous pouvez utiliser l'éditeur de votre choix. Cependant, beaucoup de compilateurs C (comme anjuta ou Kdevelop sur Linux et visual C/C++ de Microsoft) sont livrés avec un environnement de développement intégré (EDI) qui permet de créer, de compiler et d'effectuer la liaison de façon très conviviale. Consultez vos manuels pour savoir si vous possédez un tel environnement.
2. Utilisez le clavier pour saisir le code source hello.c comme indiqué dans le Listing 1.1 en appuyant sur Entrée à la fin de chaque ligne.

Les numéros de ligne de notre exemple ont été ajoutés pour une meilleure compréhension. Vous ne devez pas les introduire dans votre source.

3. Sauvegardez votre fichier source sous le nom hello.c.
4. Vérifiez que le fichier se trouve bien dans votre répertoire.
5. Exécutez la commande appropriée pour la compilation et la liaison de hello.c.
6. Contrôlez les messages envoyés par le compilateur. Si vous n'avez reçu aucun message d'erreur ou warning, votre fichier source est bon.

   Remarque : Si vous avez fait une erreur de frappe dans votre programme, comme taper `prntf` pour `printf`, le compilateur vous enverra un message comme celui-ci : `Error: undefined symbols:_prntf in hello.c (hello.OBJ)`.
7. Retournez à l'étape 2 si vous avez un message d'erreur. Éditez le fichier hello.c pour comparer son contenu avec Listing 1.1. Faites les corrections nécessaires puis passez à l'étape 3.
8. Votre premier programme C est maintenant prêt à être exécuté. Si vous faites une liste de tous les fichiers de votre répertoire qui s'appellent hello, vous allez voir apparaître :

```
hello.c qui est le fichier source que vous avez créé.
hello.obj ou hello.o qui contient le code objet de hello.c.
hello.exe ou tout simplement hello qui est le programme exécutable, résultat de la
compilation et de la liaison.
```

9. Pour exécuter hello ou hello.exe, entrez simplement `hello` sur DOS et Windows, et `./hello` sur Linux et Unix (il se peut que `hello` au lieu de `./hello` fonctionne aussi, ce qui ne serait pas anormal). Le message "Hello, world !" apparaît à l'écran.

Félicitations ! Vous venez de créer, de compiler et d'exécuter votre premier programme C.

Les erreurs de compilation

Une erreur de compilation apparaît lorsque le compilateur rencontre du code source qu'il ne peut pas compiler. Heureusement, les compilateurs d'aujourd'hui vous indiquent la nature et l'emplacement des erreurs pour faciliter la correction du code source.

Cela peut être illustré en introduisant délibérément une erreur dans hello.c. Éditez ce fichier et effacez le point-virgule à la fin de la ligne 5. hello.c ressemble maintenant au fichier du Listing 1.2.

Listing 1.2 : *hello.c* avec une erreur

```
1 : #include <stdio.h>
2 :
3 : int main()
4 : {
5 :    printf("Hello, World!")
6 :    return 0;
7 : }
```

Sauvegardez votre fichier et compilez-le. Votre compilateur va vous envoyer un message qui ressemble à celui-ci :

```
hello.c(6) : Error: ';' expected
```

Vous pouvez remarquer que cette ligne comporte trois parties :

```
hello.c              Le nom du fichier dans lequel se trouve l'erreur
(6) :                Le numéro de la ligne où a été détectée l'erreur
Error: ';' expected  Un descriptif de cette erreur
```

Le message vous indique qu'à la ligne 6 de hello, le compilateur n'a pas trouvé de point-virgule. Cette réponse étonnante vient du fait que le point-virgule que vous avez effacé ligne 5 aurait pu se trouver à la ligne suivante (même si ce n'est pas une bonne méthode de programmation). Ce n'est qu'en contrôlant la ligne 6 que le compilateur a constaté l'absence de point-virgule.

Cela illustre l'ambiguïté des messages d'erreur des compilateurs C. Vous devrez utiliser votre connaissance du langage C pour interpréter ces messages. Les erreurs sont souvent sur la ligne indiquée ou sur celle qui la précède.

Les messages d'erreur peuvent différer d'un compilateur à l'autre. Dans la plupart des cas, les indications qu'il vous fournira vous donneront une bonne idée du problème et de son emplacement.

Avant de continuer notre étude, considérons un autre exemple d'erreur de compilation. Éditez hello.c et transformez-le comme indiqué :

1. Replacez le point-virgule à la fin de la ligne 5.
2. Effacez les guillemets juste avant le mot `Hello`.

Sauvegardez le fichier et compilez de nouveau le programme. Le message d'erreur du compilateur devient :

```
hello.c(5) : Error: undefined identifier "Hello"
hello.c(7) : Lexical error: unterminated string
Lexical error: unterminated string
Lexical error: unterminated string
Fatal error: premature end of source file
```

Le premier message annonce effectivement une erreur en ligne 5 au mot `Hello`. Le message `defined identifier` signifie que le compilateur n'a pas compris `Hello` parce qu'il ne se trouve pas entre guillemets. Les messages suivants, dont nous ne nous préoccuperons pas pour le moment, illustrent le fait qu'une seule erreur dans un programme C peut quelquefois provoquer de multiples messages.

Voici ce que vous devrez en retenir : si le compilateur vous envoie plusieurs messages d'erreur, et que vous n'en trouviez qu'une, corrigez-la et recompilez votre programme. Cette seule correction pourrait annuler tous les messages.

Les messages d'erreur de l'éditeur de liens

Des erreurs en provenance de l'éditeur de liens sont relativement rares et sont généralement dues à une faute de frappe dans le nom d'une fonction appartenant à la bibliothèque C. Dans ce cas, le message suivant apparaît : `Error: undefined symbols: error message`, suivi du nom mal orthographié (précédé d'un tiret).

Résumé

La lecture de ce premier chapitre vous a certainement convaincu que le choix du C comme langage de programmation est judicieux. Il offre une bonne combinaison entre puissance, portabilité et notoriété. À ces qualités s'ajoute la possibilité d'évoluer vers le langage orienté objet C++ ou Java.

Ce chapitre a décrit les différentes étapes du développement d'un programme C. Vous devez maîtriser le cycle édition-compilation-liaison-tests ainsi que les outils nécessaires à chaque étape.

Les erreurs sont indissociables du développement d'un programme. Votre compilateur les détecte et vous envoie un message d'erreur qui en donne la nature et l'emplacement. Ces informations permettent d'éditer le code source pour le corriger. Rappelez-vous cependant que ces messages ne sont pas toujours très précis, et qu'il faut utiliser votre connaissance du C pour les interpréter.

Q & R

Q Si je veux donner mon programme à quelqu'un, de quels fichiers a-t-il besoin ?

R Le fait que le langage C soit un langage compilé est un avantage. Cela signifie que lorsque votre code source est compilé, vous obtenez un programme exécutable qui se suffit à lui-même. Pour donner Hello à tous vos amis, il suffit de leur donner l'exécutable (hello ou hello.exe). Ils n'ont pas besoin du fichier source hello.c, ni du fichier objet hello.o ou hello.obj. Ils n'ont

pas besoin non plus de posséder un compilateur C. Néanmoins, diffuser les sources (hello.c) accompagnées d'une licence libre permettra à vos amis développeurs d'améliorer Hello.

Q Faut-il conserver les fichiers sources (.c) et objets (.obj ou .o) après la création du fichier exécutable ?

R Si vous supprimez votre fichier source, vous n'aurez aucune possibilité plus tard d'apporter une modification à votre programme. Vous devriez donc le garder. Pour ce qui concerne le fichier objet, vous pouvez en obtenir une copie à tout moment en recompilant le fichier source. Vous n'avez donc pas besoin de le conserver.

La plupart des environnements de développement intégrés créent des fichiers qui s'ajoutent à ceux déjà cités ci-avant. Vous pourrez les recréer aussi longtemps que vous serez en possession du fichier source (.c).

Q Faut-il utiliser l'éditeur qui est livré avec le compilateur ?

R Ce n'est pas une obligation. Vous pouvez utiliser l'éditeur de votre choix du moment que vous sauvegardez le code source en format texte. Si votre compilateur possède un éditeur, vous devriez l'essayer et choisir celui qui vous convient le mieux. J'utilise moi-même un éditeur que j'ai acheté séparément alors que tous les compilateurs que j'utilise en ont un. Ces éditeurs qui sont livrés avec les compilateurs sont de plus en plus performants. Certains formatent automatiquement votre code source. D'autres distinguent les différentes parties de votre fichier source à l'aide de couleurs différentes pour vous aider à trouver les erreurs.

Q Peut-on ignorer les messages d'avertissement ?

R Certains de ces messages n'affectent en rien l'exécution de votre programme, d'autres pas. Si votre compilateur vous envoie un message de warning, cela signale que quelque chose ne convient pas. Beaucoup de compilateurs vous offrent la possibilité de supprimer ce genre de message en dessous d'un certain niveau. Le compilateur ne donnera que les messages les plus sérieux. Vous devriez cependant consulter tous vos messages, un programme est meilleur s'il ne comporte aucune erreur ou warning (le compilateur ne produit pas de programme exécutable s'il reste une seule erreur dans le source).

Atelier

Cet atelier comporte un quiz destiné à consolider les connaissances acquises dans ce chapitre et quelques exercices pour mettre en pratique ce que vous venez d'apprendre. Essayez de comprendre les réponses fournies dans l'Annexe I avant de passer au chapitre suivant.

Quiz

1. Donnez trois raisons pour lesquelles le C est un bon choix de langage de programmation.
2. Quel est le rôle du compilateur ?
3. Quelles sont les étapes du cycle de développement d'un programme ?
4. Quelle commande permet de compiler le programme program.c ?
5. Votre compilateur exécute-t-il la compilation et la liaison avec la même commande ?
6. Quelle extension devriez-vous utiliser pour votre fichier source C ?

7. Est-ce que filename.txt est un nom correct pour votre fichier source C ?
8. Que faut-il faire si le programme que vous avez compilé ne donne pas les résultats escomptés ?
9. Qu'est ce que le langage machine ?
10. Que fait l'éditeur de liens ?

Exercices

1. Éditez le fichier objet créé avec Listing 1.1. Ressemble-t-il au fichier source ? (Ne sauvegardez pas ce fichier lorsque vous quitterez l'éditeur.)
2. Entrez le programme suivant et compilez-le. Que fait-il ? (Ne saisissez pas les numéros de ligne ni les deux points.)

```
 1 : #include <stdio.h>
 2 :
 3 : int rayon, aire;
 4 :
 5 : int main()
 6 : {
 7 :   printf("Entrez le rayon (ex 10) : ");
 8 :   scanf("%d", &rayon);
 9 :   aire =  (3.14159 * rayon * rayon);
10 :   printf("\n\nAire = %d\n", aire);
11 :   return 0;
12 : }
```

3. Saisissez et compilez le programme suivant. Que fait-il ?

```
 1 : #include <stdio.h>
 2 :
 3 : int x,y;
 4 :
 5 : int main()
 6 : {
 7 :   for (x = 0; x < 10; x++, printf("\n"))
 8 :   for (y = 0; y < 10; y++)
 9 :   printf ("X");
10 :
11 :   return 0;
12 : }
```

4. **CHERCHEZ L'ERREUR** : Saisissez ce programme et compilez-le. Quelles sont les lignes qui génèrent des erreurs ?

```
1 : #include <stdio.h>
2 :
3 : int main();
4 : {
5 :    printf ("Regardez bien !");
6 :    printf ("Vous allez trouver !");
7 :    return 0;
8 : }
```

5. **CHERCHEZ L'ERREUR** : Saisissez ce programme et compilez-le. Quelles sont les lignes qui génèrent des erreurs ?

```
1 : #include <stdio.h>
2 :
3 : int main();
4 : {
5 :    printf ("Ce programme a vraiment ");
6 :    do_it("un problem !");
7 :    return 0;
8 : }
```

6. Transformez la ligne 9 de l'exercice 3 comme indiqué. Recompilez et exécutez le nouveau programme. Que fait-il maintenant ?

```
9 :  printf("%c", 65);
```