第6章　C言語 入門編

Chapter　1　C 言語プログラムの作成
Chapter　2　C 言語のやさしい入門

タイトル：「新訂 新 C 言語入門 ビギナー編」
著者：林 晴比古
出版：ソフトバンク パブリッシング株式会社

6.1　UNIX 環境の使い方

5.3 節 (p.77) で述べたように、**C** は **UNIX**[1] を記述するために開発された言語です。本テキストでも、Microsoft 社製の OS[2] (**以後、「Windows」と記す**) 上に UNIX 互換環境を構築する Cygwin ソフトウェア (**以後、「Cygwin」と記す**) を導入し、Cygwin のソフトウェア群に含まれるフリーで最もポピュラーな **GNU**[3] C と呼ばれる C 処理系[4]を使って C 言語を学習します。Cygwin の入手方法とインストール方法についてはインターネット[5]や書籍[6]などに詳しく解説されていますので、そちらを参照してください。なお、プログラミング演習を受講している方で、希望者には、筆者特製の Cygwin インストール CD-ROM を配布しています。

以後、
「Cygwin の UNIX 互換環境」≒「UNIX 環境」
とし、UNIX 環境に準じた解説を行います。

* Cygwin に特有なものもあるので、注意して読んでください。

[1] マルチユーザ・マルチタスク・仮想記憶方式・タイムシェアリングシステムなどのモダンな機能を備えています。現在使用されている多くのオペレーティングシステムにも、これら UNIX のモダンな機能が取り入れられています。
[2] Windows 98, Windows Me, Windows 2000, Windows XP など。
[3] 「Gnu is Not Unix」の意で、「ぐぬー」と発音します。UNIX 互換ソフトウェア群の開発プロジェクトの総称で、フリーソフトウェアの理念に従った修正・再配布自由な UNIX 互換システムの構築を目的としています。また、GNU で開発されたソフトウェアに適用されている GPL (the gnu General Public License) は、「あらゆるソフトウェアは自由に利用できるべき」という理念を実現したライセンスとして知られています。
[4] 「新訂 新 C 言語入門 ビギナー編」の 11 ページを参照してください。
[5] 「Cygwin」・「入手方法」・「インストール」をキーワードにサーチエンジンを使って検索するとよい。
[6] UNIX の入門書として数社から販売されています (Cygwin インストール CD-ROM が付属している場合が多い)。

UNIX 環境では、キーボードから文字列のコマンド (命令) を図 6.1 の**シェル**[7] (shell) に入力することでコンピュータを操作する方法が一般的です。このような環境を **CUI**[8] (Character User Interface) と呼び、コンピュータを熟知した人にとっては非常に有効なコンピュータの操作方法です。C 言語を学習する前にシェルの使い方と基本的なコマンドについて学習しておきましょう。

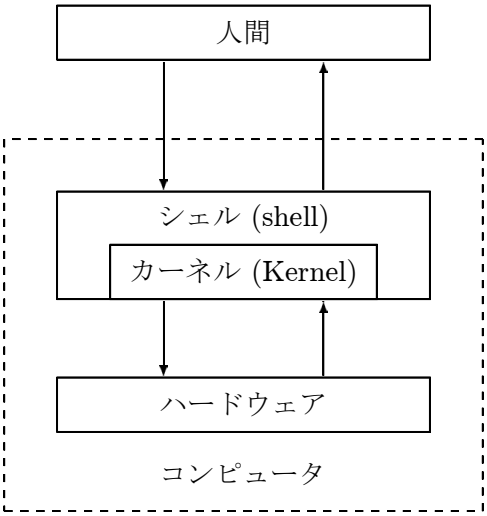

図 6.1: カーネル (Kernel) とシェル (shell)

最初に、**ファイル** (file) と**ディレクトリ**[9] (directory) について復習しておきましょう。1.1 節で述べたように、コンピュータに仕事をさせるにはプログラムとデータが必要で、これらはハードディスクなどの記憶装置に**ファイル**という形で記憶されています。これらのファイルを種類や用途ごとにまとめて入れておく箱が**ディレクトリ**で、人間にも理解しやすく、効率的なファイルの管理 (読み書き) ができる様になっています。この仕組みはオペレーティングシステムの**ファイルシステム**[10] (file system) という機能によって提供され、**木構造** (tree structure; **tree 構造**) と呼ばれる階層的な**ディレクトリ構造**を持っています。UNIX も図 6.2 のように最上位ディレクトリである「`/` (**ルート**[11])」と呼ばれる特別なディレクトリを起点に、様々なファイルやディレクトリがこのディレクトリ構造にしたがって記憶されています。なお、UNIX のディレクトリ名には古くからのしきたりがあり、「`bin`」は binary、「`lib`」は library、「`tmp`」は temporary、「`var`」は variety のように決まった略記が用いられます 。さらに、UNIX は複数のユーザが同時に同じコンピュータを利用できる機能を持っているため、通常は「`home`」の下位に各ユーザが自由に使用

[7]「殻」の意。殻 (shell) によって果実や木の実の核 (Kernel) が隠されている様子をシェルとカーネルの関係に例えて、これらの名前がつけられました。なお、Windows にも「DOS 窓」・「コマンドプロンプト」などと呼ばれる CUI が備わっています。

[8]逆に、マウスなどを使って画面に表示されている絵やアイコンなどを操作することでコンピュータを操作する環境を **GUI** (Graphical User Interface) と呼び、初心者でも直感的または視覚的にコンピュータを操作することができます。

[9]一般ユーザに親しみやすい表現として**フォルダ** (folder) という言葉も使用されています。

[10]Windows には FAT や NTFS、UNIX には EXT2 や XFS などのファイルシステムがあります。なお、ファイスシステムが提供するその他の機能としては、ファイルを作成した日時や最後に更新した日時の管理、ファイルのアクセス権限の管理などがあります。

[11]Windows では「マイ コンピュータ」が UNIX のルートに相当します。

できるディレクトリが割り当てられます。このディレクトリを**ホームディレクトリ**と呼び、例えば、図 6.2 では「Administrator」がホームディレクトリにあたります。

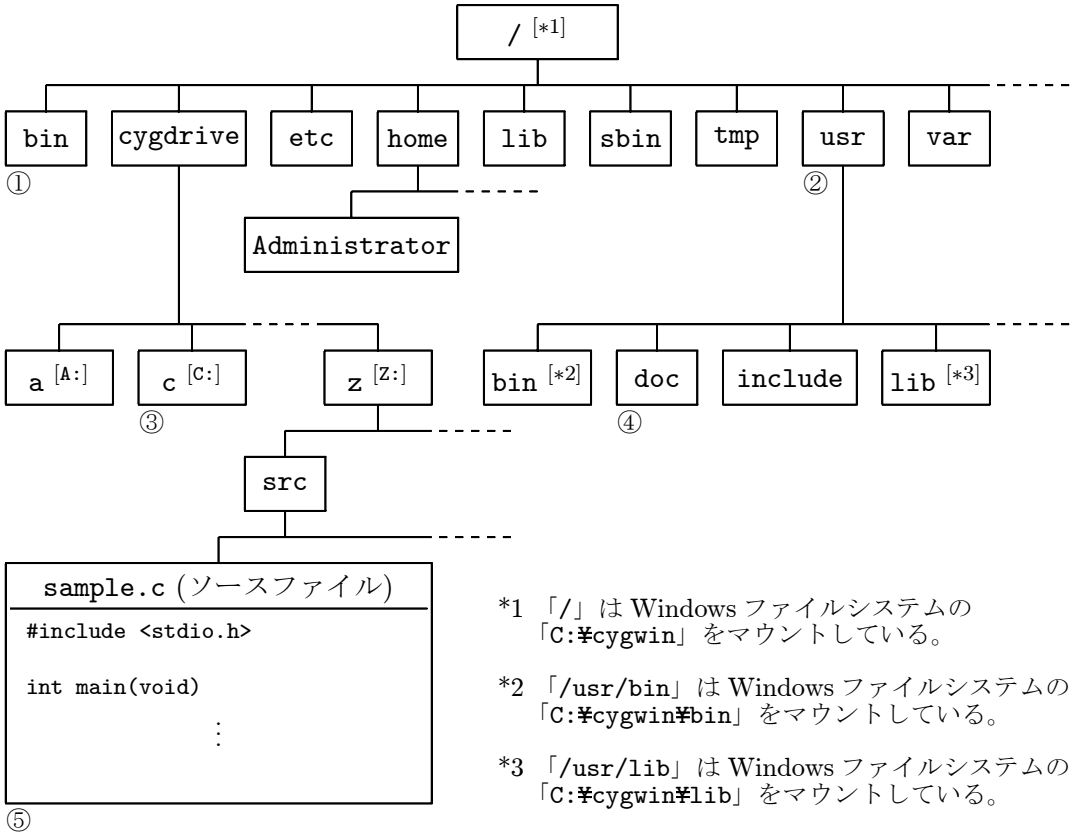

図 6.2: Cygwin (UNIX) のディレクトリ構造

なお、ディレクトリ「cygdrive」は Cygwin に特有なディレクトリで、「cygdrive」の下位ディレクトリを通して Windows のドライブへアクセスできるようになっています (例えば表 6.1)。

Cygwin のディレクトリ	Windows のドライブ
a (/cygdrive/a)	A: (フロッピディスク)
c (/cygdrive/c)	C: (ハードディスク)
d (/cygdrive/d)	D: (CD-ROM)
⋮	⋮
z (/cygdrive/z)	Z: (ネットワークドライブ)

表 6.1: Cygwin のディレクトリと Windows のドライブの対応

【注意】表 6.1 の「Z: (ネットワークドライブ)」は富山大学端末室のコンピュータに特有のドライブです。このドライブがあるおかげで、どの端末室のどのコンピュータを利用しても同じ環境でコンピュータを使用することができるようになっています。従って、授業でもこのディレクトリ内で作業 (C 言語の学習) を行います。それ以外の場合はホームディレクトリ内で作業を行うと良いでしょう。

シェル (UNIX 環境) 上では、**パス**[12] (path) と呼ばれる場所 (ディレクトリ) を指定する方法が非常に重要で、ユーザは**ディレクトリの移動**や**使用する命令・対象となるデータの場所**などを的確に指定しながらコンピュータに命令を与え作業を進めて行く必要があります。特に UNIX 環境では、特別なディレクトリを表 6.2 のような記号を使って示し、ディレクトリやファイルの区切り記号にはルートと同じ「/ (スラッシュ)」を使います[13]。なお、現在ユーザが居る場所 (ディレクトリ) を**カレントディレクトリ** (current directory) または**ワーキングディレクトリ** (working directory) と呼びます。

記号	記号の説明
/	ルートディレクトリを表す
.	カレントディレクトリを表す
..	1 つ上のディレクトリを表す
~	ホームディレクトリを表す

表 6.2: 特別なディレクトリを表す記号

パスの指定方法には**絶対パス**と**相対パス**の 2 種類があります。**絶対パス**は、最上位ディレクトリであるルートから順に下位ディレクトリをたどりながら目的のディレクトリまでを指定する方法で、必ずルートを表す記号「/」から始まります。**相対パス**は、あるディレクトリ (カレントディレクトリ) から順に上位または下位のディレクトリをたどって目的のディレクトリを指定する方法です。特別なディレクトリを表す記号や相対パスは、最終的には絶対パスに変換されて目的のディレクトリを示します。図 6.2 の場合のパスの書き方の例を以下に挙げておきます。ただし、相対パスはカレントディレクトリをホームディレクトリ「`/home/Administrator`」として示したものです。

絶対パスの例:

図 6.2 のディレクトリ① ··· `/bin` `/bin/` `/bin/.`
図 6.2 のディレクトリ② ··· `/usr` `/usr/` `/usr/.`
図 6.2 のディレクトリ③ ··· `/cygdrive/c` `/cygdrive/c/` `/cygdrive/c/.`
図 6.2 のディレクトリ④ ··· `/usr/doc` `/usr/doc/` `/usr/doc/.`
図 6.2 のファイル⑤ ··· `/cygdrive/z/src/sample.c`

相対パスの例: * いろいろな書き方ができます。

図 6.2 のディレクトリ① ··· `../../bin` `../../bin/` `../../bin/.` など
図 6.2 のディレクトリ② ··· `../../usr` `./../../usr/` `./../../../usr/.` など
図 6.2 のディレクトリ③ ··· `../../cygdrive/c` `../../cygdrive/./c` など
図 6.2 のディレクトリ④ ··· `../../usr/doc` `../../usr/../usr/doc/` など
図 6.2 のファイル⑤ ··· `../../cygdrive/z/src/sample.c` など

[12] 実は、Cygwin の起動時にプログラムが収められているディレクトリを登録するようになっていて、あたかもカレントディレクトリに命令が存在しているかのように振舞います。このことを「**パスを通す**」と呼びます。
[13] Windows の CUI では「¥」が使用されます。

では、実際に Cygwin を起動し、基本的な命令を習得しましょう。最も簡単な起動方法は Windows のディストップにある図 6.3 の Cygwin のアイコンをダブルクリックします[14]。なお、Cygwin を終了するにはウィンドウの「閉じる」ボタンをクリックします[15]。

図 6.3: Cygwin のアイコン

Cygwin を起動すると図 6.4 のようなウィンドがディスクトップ上に表示されます[16] (bash と呼ばれるシェルが起動します)。図 6.4 の各部分の解説を表 6.3 に載せておきます。

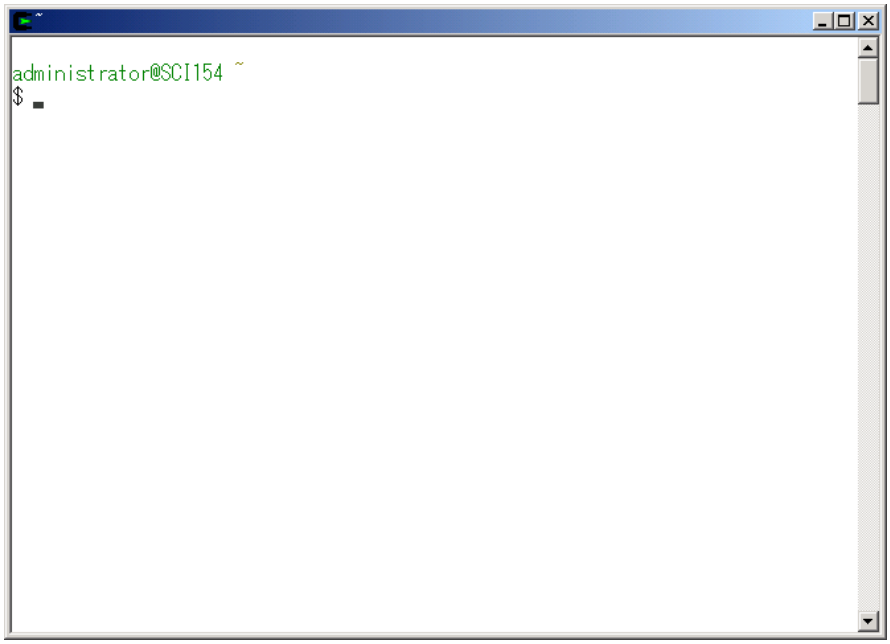

図 6.4: Cygwin のウインドウ

[14]実際には、Windows のコマンドプロンプト上でディレクトリ「C:¥cygwin」にあるバッチファイル「cygwin.bat」の記述にしたがって bash というシェルを起動しているだけです。従って、次のような方法でも起動できます。
(1) 「マイ コンピュータ」→「C:」→「cygwin」の順にディレクトリを開き、「cygwin.bat」と名前のついたアイコンをダブルクリックする。
(2) まず、メニューバーの「スタート」→「プログラム」→「アクセサリ」→「コマンドプロンプト」を順にたどり、プログラム「コマンドプロンプト」を起動する。次に、プロンプトに続けて「C:¥cygwin¥bin¥bash --login -i」と入力し、Enter キーを押す。

[15]プロンプトに続けて「exit」と入力し Enter キーを押しても終了できます。

[16]このテキストは各自で印刷することを前提としているので、印刷時にインクが少なくて済むよう背景を黒から白に、文字を白から黒に変更してあります。デフォルトは背景が黒で、文字が白となっています。

記号	記号の説明
Administrator	ユーザ名 (各ユーザごとに異なる)
SCI154	コンピュータ名 (理学部 (SCIence) 端末室のコンピュータ)
@ (アットマーク)	電子メールのアドレスと同じで、ユーザ名とコンピュータ名を区切る
~ (チルダ)	カレントディレクトリを表す (ホームディレクトリを示す)
$ (ドル)	プロンプト (prompt) と呼ばれ、この記号に続けて命令を入力する
▂ または ▉	カーソル (cursor) と呼ばれ、この位置に入力した文字が現れる

表 6.3: Cygwin のウインドウ (図 6.4) の各部分の解説

以後、Cygwin のウィンドウは下図のように角の丸い長方形で表します。なお、以下のルールに従って記述してあります。

【ルール 1】　図 6.4 の緑色の文字の部分とカーソルは省略し、プロンプト「$」のみを記述します。
* UNIX 環境のデフォルトのプロンプトは「$」です。
* ルート (root) と呼ばれるスーパーユーザのプロンプトは「#」です。

【ルール 2】　入力すべきコマンド (命令) は青文字のタイプライタ体 (abcb···) で記述します。命令の入力が完了したら Enter キーを押して実行してください (特別な場合を除き**省略します**)。

従って、以下のように作業を進めていくことになります。

```
$ コマンド 1 Enter
                    ··· コマンド 1 の結果が表示される ···

$ コマンド 2 Enter
                    ··· コマンド 2 の結果が表示される ···

                          ⋮
```

ファイルとディレクトリの操作に関するコマンドについて紹介しておきます (資料配布)。

- **カレントディレクトリの表示** ··· `pwd` (print working directory)
- **ディレクトリ内のファイル名の表示** ··· `ls` (list)
- **ディレクトリの移動** ··· `cd` (change directory)
- **ディレクトリの作成** ··· `mkdir` (make directory)
- **ディレクトリ名の変更**および**ファイル名の変更** ··· `mv` (move)
- **ディレクトリの削除**および**ファイルの削除** ··· `rm` (remove)
- **ファイルの内容の表示 (1)** ··· `more` または `cat`
  * [スペース] キーまたは [f] キー[17]で次のページを表示。 [q] キー[18]で終了。
- **ファイルの内容の表示 (2)** ··· `less`
  * [スペース] または [f] で次のページを表示。 [b] キー[19]で前のページを表示。 [q] キーで終了。

もし、コマンドの使い方を忘れたときはオプション「`--help`」を使って調べることができます。例えば、オプション「`--help`」を付けてコマンド「`ls`」を実行すると下図のようにヘルプが表示されます。もちろん、英語です。

```
$ ls --help
Usage: ls [OPTION]... [FILE]...
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuSUX nor --sort.

Mandatory arguments to long options are mandatory for short options too.
  -a, --all                  do not ignore entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               with -l, print the author of each file
  -b, --escape               print octal escapes for nongraphic characters

                                  ... 中略 ...

Exit status is 0 if OK, 1 if minor problems, 2 if serious trouble.

Report bugs to <bug-coreutils@gnu.org>.
```

[17] forward の頭文字。
[18] quit の頭文字。
[19] backward の頭文字。

また、多くのコマンドにはマニュアルが付属しているので、コマンド「man」を使用して調べることもできます[20]。例えば、コマンド「ls」を「man」で調べるには下図のように入力します。

```
$ man ls
```

画面が切り替わって下記のようなマニュアルが表示されます。

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuSUX nor --sort.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

:
```

* スペース または f で次ページを表示。 b で前ページを表示。 q で終了。

更に、シェル「bash」の便利な使い方を幾つか挙げておきます。

履歴機能　シェルにはこれまで使用したコマンドを履歴しておく機能があります。コマンドの履歴を表示するにはコマンド「history」を利用します。更に、履歴されたコマンドを再利用するには「!」に続けて「history で表示された番号」または「目的のコマンドの最初の頭文字[21]」を入力します 。使用例を以下に挙げておきます。

[20] Cygwin のマニュアルは英語ですが、日本語のマニュアルも整備されており、UNIX 環境を整備すれば利用することができます。また、インターネットや書籍で調べるのも良いでしょう。

[21] 頭文字が一致する履歴されたコマンドの内、一番最後のコマンドが選択され実行されます。

```
$ history
    1  pwd
    2  ls
    3  history

$ !1
pwd
/home/Administrator

$ !p
pwd
/home/Administrator
```

また、[↑] キーを繰り返し押すと、履歴されたコマンドが新しいものから順に表示されます。逆に、[↓] キーを押すと、1 つ新しい履歴されたコマンドが表示されます。

補完機能 補完機能を使用すると、長いファイル名やディレクトリ名を全て入力する必要がなくなります。補完機能を使用するには目的のファイル名・ディレクトリ名・コマンド名の最初の数文字を入力し、[Tab] キーを押します。使用例を以下に挙げておきます (「`mple.txt`」が補完されフィル名「`sample.txt`」が表示されます)。

```
$ ls
file.txt  sample.txt  test.txt

$ more sa[Tab]
```

⟹

```
$ ls
file.txt  sample.txt  test.txt

$ more sample.txt[Enter]
```

更に、続けて 2 回 [Tab] キーを押すと、入力した最初の数文字に該当するファイル名・ディレクトリ名・コマンド名の一覧を表示します[22]。使用例を以下に挙げておきます (1 回目の [Tab] キーで「`le`」が補完され、更に、続けて 2 回 [Tab] キーを押すことで補完された文字列「`file`」に該当するファイル名やディレクトリ名の一覧を表示します)。

```
$ ls
file1.txt  file2.txt  file3.txt

$ more fi[Tab][Tab][Tab]
```

⟹

```
$ ls
file1.txt  file2.txt  file3.txt

$ more file
file1.txt  file2.txt  file3.txt

$ more file1.txt[Enter]
```

* 最後の入力は、「`1.txt`」の代わりに「`1`[Tab]」と入力することもできる。

[22] 検索範囲は、カレントディレクトリとパスの通っているディレクトリ内にあるディレクトリとコマンドが対象となります。

出力制御記号　通常、コマンドを実行した結果は画面に出力 (表示) されますが、「| (パイプ)」や「> (リダイレクト)」(または「<」・「>> (追記)」・「<<」) を使用すると、他の命令に出力結果を渡したりファイルに出力結果を保存したりすることができます。使用例を以下に挙げておきます。

```
$ man ls | more          ← 「man ls」の出力結果を「more」によって 1 ページずつ表示
```

```
$ ls --help > file.txt   ← 「ls --help」の出力結果をファイル「file.txt」に保存
```

```
$ ls >> file.txt         ← 「ls」の出力結果をファイル「file.txt」に追加保存
```

* 単独で使用するするだけでなく、次のコマンド制御記号などと組み合わせて使用します。

コマンド制御記号　「; (セミコロン)」を使用すると 1 行に複数の命令を記述することが可能で、先頭から順にコマンドを実行します。また、「& (アンド)」を使用すると、**マルチタスク**[23]の恩恵を受けることができ、直ちに次のコマンドを入力することができます。使用方法を以下に挙げておきます。

```
$ コマンド 1 ; コマンド 2 ; コマンド 3
```

```
$ コマンド &
[1] 1959

$                                ← 直ちに次のコマンドが入力できる状態になる
```

以上のように、この節を通してシェルの使い方やコマンドの使い方を学習しましたが、これはほんのさわりでしかありません。シェルをもっと便利に使いこなすためには、「詳しい使い方」とか「多くのコマンドを知る」ということもありますが、その他にも「シェルの仕組み」・「シェルとカーネルの関係」・「環境変数」・「正規表現」・「ショートカットキー」など様々な事柄を知る必要があります。従って、一度に全てを理解しようとするのではなく、必要に応じて必要な分だけ少しずつ習得していきましょう。

自分で調べることが大切！

[23] **タイムスライス**という方法を用いて、複数の仕事をあたかも同時に行っているようにみせる機能です。

6.2 コンパイルとリンク

C 言語によるプログラミングは、以下の①から③までの作成手順を踏んで行います (図 6.6)。なお、手続き型言語の内、**コンパイラ**[24] (機械語に変換するプログラム) によってコンピュータが直接実行できる機械語に翻訳 (**コンパイル**) してから実行する言語を**コンパイラ言語**と呼び、C 言語もその 1 つです。

① テキストエディタを使って**ソースプログラム** (**原始プログラム**) を作成する。

② **コンパイラ** (翻訳プログラム) でソースプログラムを**コンパイル** (翻訳) し、**オブジェクトモジュール** (**目的プログラム**) を作成する (**オブジェクトモジュールは完全な機械語プログラムではない**)。より詳しく述べると、コンパイラは図 6.5 のように「**プリプロセス**（前処理）」→「コンパイル」→「**アセンブル**」の過程を経てオブジェクトモジュールに変換される (詳細については 7.8 節で解説します)。

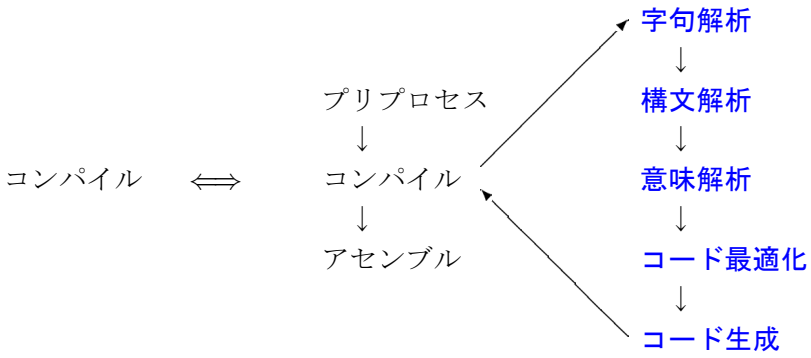

図 6.5: コンパイルの詳細

③ **リンカ** (**連結編集プログラム**) でオブジェクトモジュールと**標準ライブラリ関数** (下記参照) を**リンク** (**連結編集**) し、**ロードモジュール** (**実行形式モジュール**) を作成する (**完全な機械語プログラムになる**)。

標準ライブラリ関数　米国規格協会が標準化した ANSI/ISO 9899 (通称 ANSI C) によって定義された標準ライブラリ関数は、誰もが必要とする基本的な仕事をする関数の集まりで、これら関数によってコンピュータシステムの依存性から開放され、汎用性の高いプログラムを作成することができます。なお、標準ライブラリ関数は、各関数の定義と索引が記述された**ヘッダファイル**と各関数のオブジェクトモジュールの集合体である**オブジェクトファイル**から成っています (「新訂 新 C 言語入門 ビギナー編」の Chapter 11 および 7.6 節を参照のこと)。

* 最新の国際規格は ISO/IEC 9899:1999 (通称 C99) が制定されています (JIS 規格では JIS X 3010:2003)。

ANSI: American National Standard Institute; 米国規格協会
IEC: International Electrotechnical Commission; 国際電気標準会議
ISO: International Organization for Standardization; 国際標準化機構
JIS: Japan Industrial Standard; 日本工業規格

[24] 逆に、手続き型言語で書かれたプログラムの命令文を 1 つずつ翻訳しながら実行するものを**インタプリタ**と呼び、BASIC などが該当します。

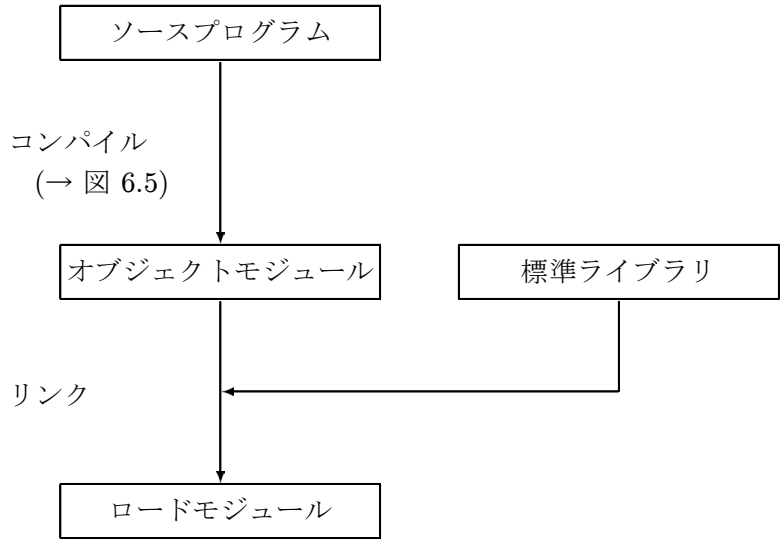

図 6.6: コンパイルとリンク

—— プログラミングを快適に行うための準備 ——

C 言語によるプログラミングを快適に行うために、以下の 3 つの準備を行いましょう。

【準備 1】ファイルの拡張子や隠しファイルが表示されるように設定を行う。

(1) ディスクトップから「マイ コンピュータ」→「コントロールパネル」の順にフォルダを開き、「フォルダ オプション」を起動する。

(2) 図 6.7 の「フォルダ オプション」のウインドウが表示されるので、

(2-1) 「表示」タブを選択する。

(2-2) 「すべてのファイルとフォルダを表示する」にチェックを入れる。

(2-3) 「登録されているファイルの拡張子は表示しない」のチェックを外す。

(2-4) 「保護されたオペレーティング システム ファイルを表示しない」のチェックを外す。

(2-5) 「OK」ボタンをクリックし、設定を反映させる。

【準備 2】カレントディレクトリ「.」にパスを通す。

(1) ディスクトップから「マイ コンピュータ」→ ドライブ「C:」の順にフォルダを開く。

(2) ファイル「autoexec.bat」を右クリックし、サブメニューから「編集」を選択する。

(3) テキストエディタ「メモ帳」が起動し、ファイル「autoexec.bat」の編集が可能となるので、

(3-1) 「set PATH=%PATH%;.」を追記する (最後のピリオドを忘れないように !)。

(3-2) 必要であれば「set CYGWIN=nontsec」を追記する (おまじない !)。

(3-3) 編集が終了したら、メニューから「上書き保存」を選択し、「メモ帳」を終了する。

* 富山大学端末室のコンピュータには、すでにこの設定がされています。

【準備 3】作業用のディレクトリ「src (source の略)」を作成する。
(1) ディスクトップから「マイ コンピュータ」→ ドライブ「Z:」開く。
(2) 新規フォルダを作成し、フォルダ名を「src」に変更する。
* 富山大学端末室のコンピュータの環境を考慮して、授業中は「Z:¥src (/sygdrive/z/src)」を作業用のディレクトリとして使用します。自宅 (自分のコンピュータ) で学習する場合は、ホームディレクトリ「/home/Administrator」に「src」というディレクトリ作成し、そこで作業するとよいでしょう。ただし、「マイ ドキュメント」[25]のように絶対パスに空白を含むようなディレクトリは避けた方が無難です。

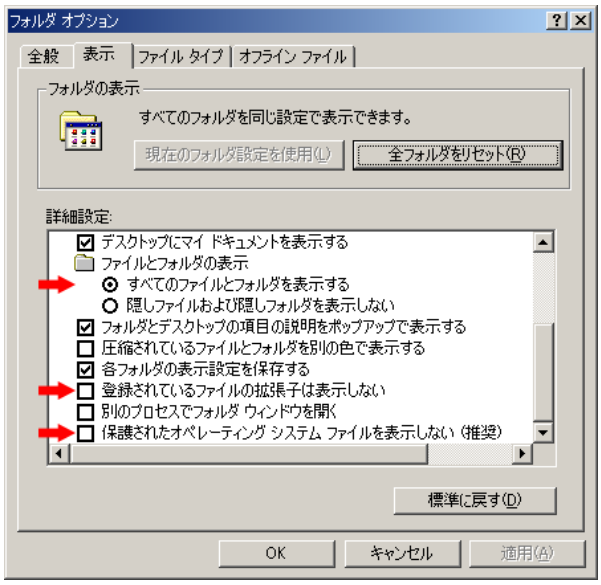

図 6.7: 「フォルダ オプション」のウインドウ

—— 初めてのプログラミング ——

簡単なプログラムを作成し、具体的なプログラミング方法を学習しましょう。

まず、Cygwin (シェル) を起動し、下記のように作業用のディレクトリに移動しておきましょう。

```
$ cd /cygdrive/z/src                    ← ディレクトリ「/cygdrive/z/src」に移動

$
```

* コマンド「pwd」などを使って、デイレクトリが移動していることを確認してください。

[25]実体のディレクトリは「C:¥Documents and Settings¥Administrator¥My Documents」です。

次に、テキストエディタを使ってソースプログラムを作成しましょう。本テキストでは、テキストエディタに「メモ帳」を使用します。図 6.8 のようにソースプログラムを正確に入力し、フォルダ「Z:¥src」にファイル名「hello.c」で保存してください。なお、この作成したファイルを「ソースファイル (source file)」と呼び、C 言語のソースファイルには必ず拡張子「.c」を付加します (逆に、拡張子が「.c」であれば、C 言語のソースファイルであることがわかります)。

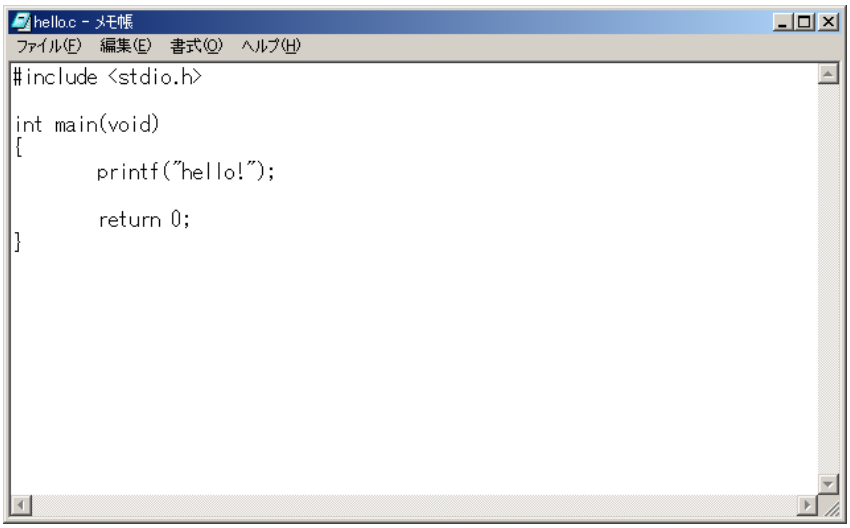

図 6.8: テキストエディタを使用してプログラムを入力

* 入力モードを「直接入力 (半角英数)」にして入力してください (**全角で入力しないこと !**)。
* 段落を揃えるには、Tab キーを使用します。

なお、本テキストでは、テキストエディタに入力すべきソースプログラムを下記のように記述します。

● 「hello」と表示するプログラム　hello.c

```
1: #include <stdio.h>
2:
3: int main(void)
4: {
5:     printf("hello¥n");
6:
7:     return 0;
8: }
```

* 「hello.c」はソースプログラムの保存に用いるファイル名です。
* 各行に付属する「行番号:」は便宜上付加したもので、実際には入力しないで下さい。
* 字下げ (インデント) は Tab キーを使用します。なお、タブの文字数は半角 4 文字とします (標準は半角 8 文字です)。

シェルからコマンド「gcc」を使ってソースプログラムをコンパイル し、ロードモジュールを作成しましょう (**コンパイル後、続けてリンクも行われます**)。下記のようにシェルにオプション「-o」を使用してコマンド「gcc」を実行してください。なお、エラーメッセージが出た場合は、入力したソースプログラムの誤りを修正し、もう一度コンパイルし直してください。

```
$ ls hello.c                                ← ソースファイル「hello.c」の確認
hello.c

$ gcc -o hello.exe hello.c      ← ソースファイル「hello.c」のコンパイルとリンク

$ ls                                        ← ロードモジュール「hello.exe」の確認
hello.c   hello.exe

$
```

* オプション「-o」を使用すると、ロードモジュールのファイル名を指定することができます。使用しない場合は、デフォルトのファイル名「a.exe」となります (本来の UNIX 環境では「a.out」)。
* 「hello.exe」の拡張子「.exe」[26]は省略することができます。

最後に、完成したロードモジュール (プログラム; コマンド) を実行してみましょう。下記のように完成したロードモジュールのファイル名「hello.exe」をシェルに入力し、実行します。

```
$ hello.exe
hello

$
```

* 「hello.exe」の拡張子「.exe」は省略することができます。

なお、カレントディレクトリにパスが通っていない (【準備 2】の設定を行っていない) 場合は、ファイルの存在するディレクトリを明示した形でファイル名を入力する必要があります。

```
$ ./hello.exe
hello
```

```
$ /cygdrive/z/src/hello.exe
hello
```

[26]拡張子「.exe」は Windows の実行可能なプログラムであることを表しています。

6.3　基本事項

「新訂 新 C 言語入門 ビギナー編」の Chapter 1 と Chapter 2 をまとめたものです。

—— Chapter 1. C 言語プログラムの作成 ——

6.3.1　C 言語とその特徴 (■ 1.1 節)

(1) 小文字でプログラムを書く。

(2) 簡潔な表現ができる。

(3) 演算子が豊富。

(4) ポインタを用いる。

(5) データ型が豊富。

(6) 関数で構成される。

(7) 構造化制御文が備わっている。

(8) プリプロセッサ付きである。

(9) 入出力処理や文字列処理は関数で行う。

(10) 特殊文字の表現が可能。

(11) 関数プロトタイプを宣言する。

6.3.2　ソースプログラムを書く (■ 1.2 節)

- ソースプログラムのファイル名の拡張子には必ず「.c」を付ける。
- ソースプログラムのファイル名に日本語などの 2 バイトコードを使用しない。

6.3.3　エラーメッセージの読み方 (■ 1.4 節)

- エラーメッセージはコンパイル時にコンパイラが見つけたソースプログラムの誤りである。
- 嘘のエラー行を知らせる場合がある。
- たった 1 つの誤りが多くのエラーを誘発する。

- よくあるエラーに慣れる (ほとんどが、単純な入力ミス)。
  - 「{」と「}」の対応が取れていない。
  - 「(」と「)」の対応が取れていない。
  - 「"」と「"」の対応が取れていない。
  - 文末の「;」の付け忘れ。
  - スペルミス。
  - 「{」と「[」(または「}」と「]」) の入力ミス。
  - 全角の空白を含んでいる (**必ず「直接入力 (半角英数)」モードで入力すること !**)

—— Chapter 2. C 言語のやさしい入門 ——

6.3.4 最初の C プログラム (■ 2.1 節)

● 「hello」と表示するプログラム (再掲) hello.c

```
1: #include <stdio.h>
2:
3: int main(void)
4: {
5:     printf("hello¥n");
6:
7:     return 0;
8: }
```

- 1 行: 「stdio.h」というヘッダファイルを利用 (■ 10.1 節を参照)。
  - # (プリプロセッサで処理)。
  - include (「含める」の意)。
  - stdio (STandarD Input Output; 標準入出力)
  - .h (ヘッダファイルの拡張子)
- 3 行: いちばん最初に実行される特別な関数「main」。
  - int 型の関数。
  - 引数はなし (void)。
- 4 行: 「{」から 8 行の「}」までが関数「main」の範囲。

- 5 行:　printf で文字列の出力。
  - 「"」から「"」までが出力範囲。
  - 「¥n」は改行を出力 (「¥」はフォントの違いで「\」で表示される場合がある)。
  - 文末には「;」を付ける。
- 7 行:　関数「main」の戻り値。
  - 「0」で正常終了。「1」で異常終了。
  - 文末には「;」を付ける。

* 字下げや空白類文字でソースプログラムを読みやすくしましょう (「Coding Standard」というソースプログラムの書き方に関するガイドラインがある)。本テキストは、「新訂 新 C 言語入門 ビギナー編」に従います。

6.3.5　数値計算をする (■ 2.2 節)

- コメントは「/*」と「*/」で囲む (「//」を使ったコメントも可能)。
- 変数は宣言してから使用する (これを怠ったエラーがよくある)。
- フォーマット付の入出力の内、数値については書式設定と変換方法の指定が必要。

6.3.6　C 言語の予約語 (■ 2.6 節)

- 「int」・「double」 (Chapter 3)。
- 「for」・「if」・「else」・「while」・「do」・「case」・「break」・「continue」 (Chapter 4)。
- 「return」 (Chapter 7)。

—— その他 ——

6.3.7　2 バイトコードに関する注意事項

- 2 バイトコードは、文字列への代入・フォーマット付の入出力・コメント以外に使わないこと。
- 2 バイトコードが文字化けする場合には文字化けするコードの前に「¥」を入れる。
- ソースファイルのファイル名に 2 バイトコードを使用しない。