3 コンパイラ g95 のインストール方法

インターネット上で公開されているコンパイラ g95 を各種の PC にインストールする方法を示す．本節の内容は，執筆時点のインストール例を示すものであり，ホームページの改変や，バージョンアップ等によるファイル名の変更等があった場合には適宜対応されたい．なお，現在 g95 は頻繁にバージョンアップが行われ，機能の向上が図られているようであるので，定期的に最新バージョンのものを取得するとよいだろう．

3.1 Linux PC への g95 のインストール

g95 を Linux PC にインストールするには，G95 のホームページ（http://g95.org/）にあるバイナリパッケージを利用するのが簡単である．ここでは，このサイトから取得したバイナリファイルを Linux PC（Linux 2.6.0, gcc-3.3.1-5）にインストールする例を示す．

まず，上記サイトからバイナリファイル g95-x86-linux.tgz をダウンロードし，適当なディレクトリ（例えば/tmp）に保存する．スーパーユーザとなり，以下のようにして /usr/local/g95 というディレクトリを作成し，ダウンロードしたファイルをそこへ移動して展開する（以下，行頭の # と % は，それぞれスーパーユーザと一般ユーザのプロンプトを表す．これらを入力する必要はない）．

```
# mkdir /usr/local/g95
# cd /usr/local/g95
# mv /tmp/g95-x86-linux.tgz .
# tar xvzf g95-x86-linux.tgz
```

すると，/usr/local/g95 以下に，g95-install というディレクトリが作られる．このディレクトリ内には，その中にコンパイルを行うための実行ファイルを含んだ bin ディレクトリや，インストールの方法が書かれたテキストファイル，pdf 形式の詳細な利用マニュアルなどが作られる▼15．バイナリファイルを展開したディレクトリへ移動して，以下のようにパスが通してあるディレクトリ（以下の例では/usr/local/bin）にシンボリックリンクを張る．

```
# cd /usr/local/g95
# ln -ivs $PWD/g95-install/bin/*g95* /usr/local/bin/g95
```

上記により，/usr/local/bin ディレクトリ内に g95 という名称で実行ファイルのシンボリックリンクが作成される．

以上で g95 のインストールは終了である．新しくシェルウインドウを開き，プログラムを作成する適当なディレクトリ（例えば f90src 等）を作成して，そこへ移動する．

▶15 bin ディレクトリ内にある，ファイル名に g95 という語を含むファイルが実行ファイルである．

```
% cd
% mkdir f90src
% mkdir f90src/hello
% cd f90src/hello
```

そして，Emacs などのエディタを以下のようにして起動し，

```
% emacs hello.f90 &
```

テストラン用の以下の 3 行のプログラムを作成する．

◆リスト 1　hello と表示するテストプログラム

```
program hello
  write(*, *) 'hello'
end program hello
```

作成したプログラムを hello.f90 という名前のテキストファイルとして保存する▼16．次に，シェルウインドウに移り，以下のようにしてプログラムをコンパイルする．

```
% g95 hello.f90
```

コンパイルが正常に行われれば，通常何もメッセージは表示されず，再びプロンプトが表示される▼17．コンパイルに成功すると，同じディレクトリ内には a.out という実行ファイルが作成されるので，ls コマンド等で確認してみよう．a.out を実行するには，シェルウインドウ内で次のように入力する▼18．

```
% ./a.out
```

シェルウインドウ内に hello と表示されれば，リスト 1 のプログラムは正しく実行されたことになる．以上で，g95 の設定は完了である．

3.2 Apple PC への g95 のインストール

Apple PC（Mac OS X）へ g95 をインストールする方法を示す．Mac OS X は UNIX をベースとするオペレーティングシステムであり，計算環境は，Linux PC と類似する点

▶16 ファイルの名称は任意であるが，Fortran 90/95 のプログラムの拡張子は .f90 とするとよい．

▶17 もし，g95 というコマンドがない，と表示された場合には，g95 のインストールあるいはパスの設定が正しく行われていない可能性があるので確認されたい．また，タイプミスなどのプログラムの誤りがある場合には，コンパイルエラーが出るので，エディタでプログラムを修正して保存する．

▶18 カレントディレクトリ（現在作業を行なっているディレクトリ）にパスを通す設定であれば，先頭の ./ は省略可能である．

が多い．また，購入時に付属するインストールディスクの中に納められているXcodeというソフトウエアをインストールすれば，余計な出費をすることなくUNIX開発環境を整えることができる[▼19]．

Apple PCでg95を使用するには，このXcodeをインストールした後，バイナリファイルをダウンロードして簡単な設定をするだけでよい．以下に，iMac（Mac OS X 10.4.7, 2 GHz Intel Core Duo）にg95をインストールする例を示す．管理者ユーザとしてApple PCにログインしてから以下の作業を進めること．

1. 購入時に付属するMac OS Xのインストールディスクの中に，Xcode Toolsという名前のフォルダがある．このフォルダの中にあるXcodeTools.mpkgというファイルをダブルクリックすると，Xcode Toolsのインストーラが起動し，インストール作業が開始される．その後は，一般のアプリケーションのインストール方法と同様の手順を取ればよい．
2. G95のホームページhttp://www.g95.org/にアクセスし，x86 OSX(g95-x86-osx.tgz)をダウンロードする[▼20]．ダウンロードした圧縮ファイルg95-x86-osx.tgzをホームディレクトリ（ユーザ名が付けられた，家の形のアイコンを持つフォルダ）に保存する．
3. 「アプリケーションフォルダ」の中の「ユーティリティ」フォルダにあるターミナル.appを起動する．これは，UNIXのシェルウインドウに相当する．このターミナル内で次のように入力して圧縮ファイルを展開する（以下では$はプロンプトを表し，入力する必要はない）．

```
$ tar xvzf g95-x86-osx.tgz
```

4. 展開すると，g95-installというフォルダが作成される[▼21]．その中のbinフォルダに移動し，binフォルダ内にある実行ファイルをg95という名称のファイルとして，/usr/binへコピーする．

```
$ cd g95-install/bin
$ sudo cp *g95* /usr/bin/g95
```

上記の2行目を入力した際にパスワードを尋ねられたら，Apple PCにログインしたときと同じ管理者のパスワードを入力する[▼22]．

5. 次に，1つ上のフォルダに戻り，その中のlibフォルダに移動する．そして，libフォルダ内にあるフォルダ（gcc-libフォルダ）を/usr/libへ移動する．

```
$ cd ../lib
$ sudo mv * /usr/lib
```

上部のメニューバーの「ファイル」から「新規シェル」を選んで新しいターミナルウインドウを開き，which g95と入力すると，/usr/bin/g95という応答があることを確認する．

次に，テスト用プログラムを作成して，動作を確認しよう．まず，ホームディレクトリの中にf90srcというフォルダを作成し，さらにその中にhelloフォルダを作る．そして，helloフォルダ内でリスト1のプログラムを作成し，hello.f90という名前のテキストファイルとして保存する．プログラムを編集するエディタとしては，Emacsと同様に動作するAquamacs（http://aquamacs.org/index.shtml）やGNU Emacs for Mac OS X（http://www.porkrind.org/emacs/）等が使いやすい[▼23]．

プログラムを作成した後，ターミナルを起動し，次のようにhelloフォルダに移動してコンパイルする．

```
$ cd f90src/hello
$ g95 hello.f90
```

コンパイルが正常に行われると，同じフォルダ内にa.outという実行ファイルが作られる．コンパイルエラーが表示されたら，プログラムを修正して保存し，再びコンパイルを行う．a.outを実行するには，ターミナル内で次のように入力すればよい．

```
$ ./a.out
```

その結果，helloと表示されれば，計算環境は適切に設定されたことになる．

3.3 Windows PCへのg95のインストール

g95をWindows PCにインストールする方法を示す．以下は，OSとしてMicrosoft Windows XP（Home Edition ver.2002）を使用した場合の例である．

1. Internet Explorerなどのブラウザで G95のホームページ http://www.g95.org/にアクセスし，G95のバイナリファイルの一覧を表示させる．

►19 Mac OS Xの低いバージョンでは，Xcode は Developer Toolsといわれる．

►20 Intel MACではなく，PowerPC MACの場合には，Powerpc OSX (g95-powerpc-osx.tgz)をダウンロードする．以下，PowerPC MACでは，g95-x86-osx.tgzをg95-powerpc-osx.tgzと読み替える．

►21 g95-installフォルダにある，コンパイルオプション等が解説されたpdfファイルを参照するとよい．

►22 sudoにより，コマンド単位の一時的なルート権限が得られる．一度パスワードを入力すると，一定の時間パスワードなしでsudoコマンドが使える．

►23 Aquamacsは Universal Binaryといわれ，PowerPC MACと Intel Macの両方で使用できる．また，GNU Emacs for Mac OS Xは Intel MAC専用である．

② その中から Self-extracting Windows x86 ファイル（g95-MinGW.exe）を選択し，デスクトップ上などの適当な場所に保存する．

③ g95 をインストールする場所を定める．例えば，「マイコンピュータ」からローカルディスク（C:）をクリックし，その下に g95 というフォルダを新規作成する．

④ ダウンロードした g95-MinGW.exe を実行（ダブルクリック）する．インストールするフォルダを尋ねられたら，一覧表示（Browse）して上記で作成した C: の下の g95 というフォルダを選択して，Install をクリックする．

⑤ 以下，問い合わせのウインドウが示されたら，基本的には OK を選択していけばよい．インストールが適切に終了すると，コンパイラに相当する実行ファイル g95.exe が C:¥g95¥bin に作成され，パスが通されてどこからでも実行できる状態となる．コマンドプロンプトを起動して[▼24]，g95 と入力すると，

```
g95: no input files
```

と表示されれば，インストールは成功である．

⑥ g95 を削除する場合，あるいは設定を最初からやり直すときには，C:¥g95 内に作成された uninstall-g95.exe を実行する．

上記のインストールが完了したら，テスト用のプログラムと，それをコンパイル・実行するための簡単なバッチファイルを以下のようにして作成しよう．

① まず，プログラムを作成するフォルダを定める．一例として，C: の下に f90src というフォルダを作成し，さらに，その中に hello というフォルダを作成する．

② 上記の hello フォルダの中に，次の内容のテキストファイルを作成する[▼25]．

```
g95 hello.f90 && a.exe
pause
```

テキストファイルを作成するには，例えばメモ帳[▼26]などのテキストエディタを利用すればよい．このテキストファイルを cmp という名称で保存する．

③ 上記のテキストファイル cmp の拡張子は .txt となっているので，これを .bat と変更する[▼27]．

▶24 「スタート」→「すべてのプログラム」→「アクセサリ」→「コマンドプロンプト」を選択すると，コマンドプロンプトが起動する．

▶25 記号 && は，コンパイルが成功した場合に限り，作成された a.exe を実行するためのものである．コンパイルが失敗した場合には，a.exe は実行されない．また，pause は結果が表示されるウインドウがすぐに閉じないようにするためのコマンドである．

▶26 メモ帳は，「スタート」→「すべてのプログラム」→「アクセサリ」→「メモ帳」を選択すると起動する．また，Windows でも Emacs 系のエディタを利用できるようなので試してみるとよい．

▶27 拡張子を表示するには，メニューバーの「ツール」から「フォルダオプション」を選択し，「表示」タグを選んで，詳細設定の中から「登録されている拡張子は表示しない」という欄のチェックを外せばよい．

④ 一方，同じ hello フォルダ内で，メモ帳などのテキストエディタを使用して，リスト 1 に示した内容のテスト用プログラムを作成し，hello.f90 という名前のテキストファイルとして保存する．拡張子は必ず .f90 とすること．

⑤ 先に作成した cmp.bat ファイルを実行（ダブルクリック）する．すると，プログラム hello.f90 のコンパイルが行われ，プログラムに問題がなければ実行ファイル a.exe が作成される．プログラムにタイプミスなどがあると，コンパイルエラーが表示されるので，テキストエディタで hello.f90 を開いて修正してから保存し，再び cmp.bat ファイルを実行する．コンパイルが成功すると，続いて a.exe が実行され，結果を表示するウインドウの中に hello と表示される．このような結果が得られれば，プログラムをコンパイル・実行する環境は正しく設定されたことになる．

なお，異なるプログラム（例えば bye.f90）を作成して，それをコンパイル・実行する場合には，上記のバッチファイルの内容を次のように書き換える必要がある．

```
g95 bye.f90 && a.exe
pause
```

このバッチファイルの書き換えを忘れると，以前に作成した hello.f90 のコンパイルと実行が行われることになるので注意しよう．

バッチファイルを使わない場合には，コマンドプロンプトで以下のように順に入力すれば，コンパイルと実行が行える．

```
> g95 hello.f90
> a.exe
```

4 グラフ描画ソフト gnuplot のインストール方法

グラフ描画のためのフリーウエアは数多くあるが，gnuplot は手軽に使える便利な描画ツールであり，機能も豊富である．ここでは gnuplot を各種 PC にインストールする方法と簡単な使い方を紹介する．gnuplot の詳細に関しては，解説書[15] やインターネット上の情報を参照されたい．

4.1 Linux PC への gnuplot のインストール

gnuplot の ver.4.0 以降では，カラー表示やマウス操作がサポートされるなどの新機能が盛り込まれている．Linux PC にはすでに gnuplot がインストールされている場合が多いと思われるが，使用している gnuplot のバージョンが低ければ，gnuplot のホームページ http://gnuplot.info/ を参照して，以下のようにして最新版を入手するとよい．

Linux PC に gnuplot をインストールするには，上記のサイトから download の箇所へ

移動し，圧縮されたソースファイル gnuplot-4.2.3.tar.gz を取得して，適当なディレクトリに保存する．そして，以下のようにして圧縮ファイルを展開し，作成されたディレクトリ gnuplot-4.2.3 内へ移動する．

```
% tar xvzf gnuplot-4.2.3.tar.gz
% cd gnuplot-4.2.3
```

ディレクトリ gnuplot-4.2.3 内には，インストール方法の詳細が書かれたテキストファイル INSTALL.gnu があるので，内容を確認するとよい．続いてディレクトリ gnuplot-4.2.3 内で以下のように入力してコンパイルを行う[▼28]．

```
% ./configure  CC = cc
% make
```

上記の後，スーパーユーザとなって次のように入力すると，gnuplot がインストールされる．

```
% su
Password: (スーパーユーザのパスワードを入力)
# make install
# make clean
```

パスが通っていれば，新しいシェルウインドウを開いて，

```
% gnuplot -V
```

と入力すると，gnuplot のバージョンなどの情報が表示される．コマンドがないと表示されたり，旧バージョンの gnuplot が起動する場合には，パスの設定等を確認する．

4.2 Apple PC への gnuplot のインストール

gnuplot を Apple PC（Intel MAC および PowerPC MAC）にインストールする方法を以下に示す．gnuplot などの UNIX のアプリケーションを Apple PC にインストールするために，Fink という便利なツールがあるので，最初にこれをインストールする．

1. まず，3.2 項で示したように，gnuplot をインストールする前に Xcode をインストールしておく．
2. 次に，Fink のホームページ（http://fink.sourceforge.net/）のダウンロードページにアクセスして，Fink のバイナリファイルをダウンロードする．Intel MAC 用と PowerPC MAC 用のバイナリファイルがあるので，自分の PC に合うものを選ぶ．Fink のホームページには，インストール方法が詳しく示されているので，それに従い Fink の設定を行う．
3. インストール後，アプリケーションフォルダ中に FinkCommander フォルダが作成されるので，その中にある FinkCommander.app を実行する．すると，ソフトウエアの一覧が表示されるので，リストの中から gnuplot を選択する．そして，上部のメニューバーから「Binary」→「Run in Terminal」→「Insall」を選ぶとターミナルが自動的に開いてインストールが開始される．パスワードを求められたら，Apple PC の管理者のパスワードを入力する．
4. インストールが終了したら，そのままターミナル内で，

```
$ gnuplot
```

と入力すると gnuplot が起動し，gnuplot のプロンプトが表示される．そのプロンプトに以下のように入力して，グラフが表示されることを確認する．

```
gnuplot > plot sin(x)
```

以上で Apple PC における gnuplot の設定は終了である．

4.3 Windows PC への gnuplot のインストール

gnuplot を Windows PC（Microsoft Windows XP Home Edition ver.2002）にインストールする手順を以下に示す．

1. ftp://ftp.gnuplot.info/pub/gnuplot/gp400win32.zip から Windows 用の gnuplot 実行ファイルを含む圧縮ファイル gp400win32.zip をダウンロードして，例えば C: の下に保存する．この圧縮ファイルを展開するには，例えば圧縮ファイルを右クリックして「すべて展開」を選択し，「圧縮フォルダの展開ウィザード」を開始すればよい．
2. 「展開ウィザード」で展開先に C:¥ を指定すると，C: の下に gnuplot フォルダが作成され，さらにその中に bin フォルダが作られる．この bin フォルダ内にある実行ファイル wgnuplot.exe を実行（ダブルクリック）すれば gnuplot が起動する．起動するとウインドウが開くので[▼29]，その内に表示される gnuplot> というプロンプトに続けて，例えば以下のように入力するとグラフが表示される．

```
gnuplot> plot sin(x)
```

▶28 gcc が適切にインストールされていれば，./configure の後の CC = cc というオプションは不要．

▶29 gnuplot を起動して表示されるウインドウ内の文字が判別しにくい場合には，そのウインドウ内で右クリックして「Choose Font..」を選択し，MS ゴシックなどの適当なフォントを選ぶ．さらに，もう一度右クリックして「Update wgnuplot.ini」を選択し，次回も同じフォントで起動するように初期設定を変更しておく．

③ 次に，どこからでも wgnuplot.exe を起動できるように，パスを設定する[▼30]．

④ 上記の手順が終了したら，gnuplot のデモンストレーションを表示してみよう．gnuplot をインストールすると，デモンストレーション用のファイルを含んだ demo フォルダが gnuplot フォルダ内に作られる．これを表示するには，コマンドプロンプトを起動して，

```
cd C:¥gnuplot¥demo
```

と入力して demo フォルダへ移動し，

```
wgnuplot all.dem
```

と入力する．3 次元の表示などが示されたら，マウス操作を試してみよう．デモンストレーションではいろいろなグラフ表示を見ることができるが，実際の計算結果に対してこれらのグラフ表示を行うには，demo フォルダ内にある各種のファイルを参考にして，これを自分のデータ用に書き換えてやればよい．

4.4 gnuplot の簡単な使い方

gnuplot には多くの機能があるが，ここではファイルに書き出された数値を使って 2 次元あるいは 3 次元のグラフを書いたり，ベクトルを描画する方法を簡単に示す．

次のような数値データを含むテキストファイル data.d が用意されているとする．同一行の数値と数値の間には，1 つ以上のスペースあるいはタブが入っているとする．

```
# x     y     u     v
2.2   6.4   1.2   0.7
3.3   8.7   2.2   0.5
4.4   9.8   3.2   0.1
5.5   9.8   3.2  -0.1
6.6   8.7   1.2  -0.5
7.7   6.4   2.2  -0.7
```

上記のデータの 1 行目のように，# で始まる行は，gnuplot ではコメント行と見なされ，その内容は無視される．2 行目以降の 4 カラムの数値データが描画の対象となる．

最初に，data.d を含むディレクトリ内において，gnuplot と入力して[▼31]，gnuplot を起動する．すると，次のように gnuplot のプロンプトが表示される．

```
gnuplot>
```

このプロンプトの後に，次のように入力すると，1 カラム目のデータを横軸，2 カラム目を縦軸としたグラフが描画される．各点がシンボルで示され，それらが直線で結ばれて表示されるだろう．

```
gnuplot> plot 'data.d' with linespoints
```

プロンプト gnuplot> の後で q と入力すれば，gnuplot が終了する．

データを直線で結びたくない場合には with points，各点のシンボルを表示しない場合には with lines とすればよい．他のカラム，例えば 2 カラムと 4 カラムのデータを表示するには次のようにする．

```
gnuplot> plot 'data.d' using 2:4 with linespoints
```

同様に，using 4:2 とすれば，縦軸と横軸を入れ替えたグラフが表示される．

次に，data.d をコピーして data2.d とし，これをエディタで開いて，空行を 1 行入れてみる．

```
# x     y     u     v
2.2   6.4   1.2   0.7
3.3   8.7   2.2   0.5
4.4   9.8   3.2   0.1

5.5   9.8   3.2  -0.1
6.6   8.7   1.2  -0.5
7.7   6.4   2.2  -0.7
```

上記の data2.d に対して，

```
gnuplot> plot 'data2.d' with linespoints
```

としてグラフ表示を行うと，数値データのうち，3 行目と 4 行目の点の間に直線が引かれず，2 本の折れ線グラフが描かれる．このように，同一ファイル中のデータを直線で結びたくないときには，その間に空行を入れればよい．

今度は，data.d のデータを 3 次元的に表示してみよう．gnuplot のプロンプトに続いて次のように入力すると，(x, y, z) 空間中に (x, y, u) をプロットした 3 次元グラフが表示される．

▶30 パスを設定するには，「マイコンピュータ」を右クリックしてプロパティを表示し，「詳細設定」の中の「環境変数」をクリックする．ユーザー環境変数の中に PATH がない場合には「新規」をクリックし，変数名に PATH と入力して，変数値の部分に wgnuplot.exe を含むフォルダの場所を記述する．上記の例では，変数値の部分に C:¥gnuplot¥bin と記述すればよい．一方，ユーザー環境変数に PATH がある場合には，これを選択した後「編集」をクリックする．PATH に対する既存の変数値の末尾にセミコロン「;」を入れて区切り，その右側に上記のように wgnuplot.exe を含むフォルダの場所を記述する．

▶31 Linux PC ではシェルウインドウ，Apple PC ではターミナル，Windows PC ではコマンドプロンプトでこのように入力する．また，Windows PC では wgnuplot と入力して gnuplot を起動する．以下の記述でも同様である．

```
gnuplot> splot 'data.d' with linespoints
```

gnuplot のバージョン 4.0 以上ではマウス操作がサポートされているので，グラフ上でマウスをドラッグして表示を変えてみるとよい[▼32]．data.d の 4 カラム目のデータを z 軸方向にプロットする場合には，次のように入力すればよい．

```
gnuplot> splot 'data.d' using 1:2:4 with linespoints
```

次に，1，2 カラムのデータを始点とし，3，4 カラムのデータを要素とする 2 次元ベクトルを表示してみよう．gnuplot のプロンプトに続いて次のように入力する．

```
gnuplot> plot 'data.d' with vectors filled
```

上記のように末尾に filled を付けると，ベクトルの矢尻が塗りつぶされる．data.d の内容を変更せずにベクトルの長さを変えて表示するには，次のような方法がある．

```
gnuplot> plot 'data.d' using 1:2:(0.5*$3):(0.5*$4) with vectors
```

このようにすると，表示されるベクトルの長さが半分になる．また，先に描いたグラフとベクトルを合わせて表示するには，

```
gnuplot> plot 'data.d' with vectors, 'data.d' with linespoints
```

とすればよい．

このコマンドを打ち込んだ後，次のように入力すると，グラフの設定情報が script という名称のスクリプトファイル（描画のためのコマンド等が記述されたテキストファイル）として保存される．

```
gnuplot> save 'script'
```

ここで gnuplot> のプロンプトの後で q と入力し，一旦 gnuplot を終了しよう．すると，ディレクトリ内に script という名前のテキストファイルが作成されている．スクリプトファイルはテキストファイルなので，エディタで開いて編集することができる．このスクリプトファイルの末尾付近には次のような記述がある[▼33]．

```
...
plot 'data.d' with vectors, 'data.d' with linespoints
#    EOF
```

▶32 Apple PC では，X11 を起動後，xterm 内で gnuplot を起動すれば，マウス操作が可能となる．
▶33 # EOF はファイルの末尾（end of file）を意味する．

これは，スクリプトファイルを保存する前に，プロンプト gnuplot> に入力したコマンドである．script ファイルのこの部分を次のように書き換えて保存する．# とその右側の説明文は入力する必要はない．

```
...
set grid                # グリッドを表示
set size 0.75, 1        # グラフの横・縦の長さをそれぞれ 0.75，1 とする
set xrange [0 : 10]     # 横軸の範囲を 0 から 10 と指定
set yrange [5 : 10]     # 縦軸の範囲を 5 から 10 と指定
set xlabel "x"          # 横軸のタイトルを x とする
set ylabel "y"          # 縦軸のタイトルを y とする
set noclip              # グラフの軸と交わる曲線やベクトルを描かない
set format x '%4.1f'    # 横軸の数値のフォーマットを設定
set format y '%4.1f'    # 縦軸の数値のフォーマットを設定
plot 'data.d' with vectors filled title "vector", \
     'data.d' with linespoints title "x-y" pointsize 5 linewidth 1
#    EOF
```

最終行（# EOF がある行）の 1 つ上の行は，その上の行からの継続行である．このように，記述が継続する場合には，先行する行の行末にバックスラッシュ「\」を付ける．各行の命令の意味は右側のコメントのとおりである．再び gnuplot を起動して，次のようにしてこのスクリプトファイルを読み込む．

```
gnuplot> load 'script'
```

すると，スクリプトファイルで指定した表示が行われるのが確認できるだろう．以降は，スクリプトファイルを編集して上記のようにロードすることとすれば，gnuplot> のプロンプトの後で毎回コマンドを打ち込む手間が省ける．

最後に，ディスプレイ画面に表示されたグラフを，例えば LaTeX で利用する eps 形式のファイルとして保存する方法を示す．まず，次のような内容のテキストファイルをエディタで作成しておく[▼34]．

```
set terminal postscript eps enhanced color "Helvetica" 24
set output "test.eps"
replot
set terminal x11
```

作成したテキストファイルの名称が outeps であるとする．gnuplot を起動してディスプレイ画面にグラフの表示を行い，描画を行った後に，次のようにして outeps をロードする．

▶34 このスクリプトは，Linux PC あるいは Apple PC で X11 を起動した後のターミナル（xterm）で gnuplot を起動した場合に使用できる．Apple PC で X11 を起動せずに gnuplot を起動した場合，また Windows PC の場合には，4 行目の set terminal x11 は不要である．

```
gnuplot> load 'script'
gnuplot> load 'outeps'
```

outeps は，eps ファイルへの書き出しを行うスクリプトファイルとなっており，上記の 2 行目の load を行った後には，test.eps という名称でグラフに対する eps 形式の画像ファイルが生成されている．このようにして，gnuplot で作成したグラフを画像ファイルとして取り扱うことができる．

5 コンパイルの方法に関する補足

5.1 make コマンドを利用する複数のファイルのコンパイル

プログラムが複数のソースファイルから構成されるときには，UNIX PC では，make コマンドを使用するとコンパイル作業を効率的に行うことができる．make の基本的な動作原理は，複数のソースファイル（拡張子が .f90 のファイル）があるとき，それらのタイムスタンプ（最後に作成・編集された日付や時刻）を関連するオブジェクトファイル（拡張子が .o のファイル）のタイムスタンプと比較して，もしソースファイルの方が新しいか，あるいはオブジェクトファイルが存在しない場合には，そのソースファイルをオブジェクトファイルに変換する，というものである．関連するオブジェクトファイルのタイムスタンプの方が新しければ，コンパイルは行われない．このため，不必要なコンパイルが省けるので，コンパイル作業に要する時間を短縮できるというのが make の利点である．

make には豊富な機能があり，これを解説する図書も複数出版されている[16], [17]．紙面の都合上，ここではごく基本的な Makefile の例を示すこととし，記述の詳しい規則等については触れない．より進んだ使い方に興味がある読者は，参考書やインターネット上の情報を参照されたい．

ここでは，4.6 節で述べた「外部副プログラム型」あるいは「混在型」の構成を対象とし，Linux PC（Linux 2.6.0, gcc-3.3.1-5, GNU Make 3.80）で make コマンドを利用する例を示す．

- プログラムは，1 つの主プログラムと複数の外部副プログラムから構成される．
- グローバル変数モジュールを含むファイルと，インターフェイス・モジュールを含むファイルを使用する（「混在型」の場合には，モジュール副プログラムはグローバル変数モジュールの中に含まれているものとする）．

主プログラムを main.f90, 外部副プログラムを含む 2 つのファイルを sub1.f90, sub2.f90 とし，グローバル変数モジュールとインターフェイス・モジュールを含むファイルをそれぞれ globals.f90 および interface.f90 とする．これら 5 つのファイルからプログラムが構成されているとき，これらの依存関係を次のように設定する．

- globals.f90 あるいは interface.f90 に含まれるモジュールの内容が変更されたときには，すべてのファイルをコンパイルし直す．
- main.f90, sub1.f90 あるいは sub2.f90 の各ファイルの内容が変更されたときには，修正されたファイルに対するオブジェクトファイルのみを作成する．
- 上記のコンパイルを行った後，実行ファイル go を作成する．

このような条件では，次のような簡単な Makefile の利用が考えられる．ここでは，コンパイラとして g95 を用い，コンパイル時のオプションは，-fimplicit-none -fbounds-check として，デフォルトで implicit none 宣言が行われる状態とし，さらに実行時に配列の添字が上下限を越えていないかというチェックを行うとする．

```
TARGET = go
OBJECTS = interface.o globals.o main.o sub1.o sub2.o
F90 = g95
FFLAGS = -fimplicit-none -fbounds-check
COMMON_MOD = interface.f90 globals.f90

.SUFFIXES :
.SUFFIXES : .o .f90
.f90.o:
${F90} -c $< ${FFLAGS}

${TARGET} : ${OBJECTS}
${F90}  -o $@ ${OBJECTS}

${OBJECTS} : ${COMMON_MOD}
```

上記のような内容のテキストファイルをエディタで作成し，名称を Makefile あるいは makefile として保存する．この Makefile の 1 行目から 5 行目までは，マクロの定義であり，この部分を実際のファイル構成に合わせて書き換えればよい．これらの行では，等号 = の左辺に書かれた変数に右辺の記述が代入される．例えば F90 = g95 と定義すれば，それ以降の行で ${F90} と書かれた部分では，変数の内容が展開され，g95 に置き換えられる．このため，1 行目から 5 行目までの右辺を各自の環境に合わせて，次のように記述すればよい．

```
TARGET = [実行ファイルの名称]
OBJECTS = [モジュールのファイル名.o] [主・副プログラムのファイル名.o]
F90 = [コンパイラを起動するコマンド]
FFLAGS = [コンパイルのオプション]
COMMON_MOD = [モジュールのファイル名.f90]
(以下変更不要)
```

2 行目の OBJECTS = の右辺には，ソースファイル名の拡張子を .o としたものをすべて列挙する．このとき，モジュールを含むファイル名を先に書く必要がある．もし，1 行に書ききれない場合には，行末にバックスラッシュを使って次のように記述する．

```
OBJECTS = interface.o globals.o \
main.o sub1.o sub2.o
```

また，行頭にスペースを入れると誤った記述とされる場合がある▼35．

```
OBJECTS = interface.o globals.o \
(スペースキー入力)  main.o sub1.o sub2.o     <- エラーとなる場合がある
```

Makefile の 3 行目の右辺には，コンパイラを起動するコマンドを書く．例えば，コンパイラとして gfortran を使う場合には，上記 Makefile の 3 行目を F90 = gfortran とすればよい．また，4 行目の右辺にはコンパイル時のオプションを書く．オプションが何もなければ，

```
FFLAGS =
```

のように，= のすぐ後で改行して，何も書かずに空のマクロとしておく．5 行目の COMMON_MOD = の右辺には，モジュールファイル名を書く．ここに書いたファイルの内容が変更されると，2 行目の右辺に書いたすべてのファイルが再コンパイルされる▼36．実際のプログラムでは，モジュールを使用しない外部副プログラムから構成されるファイルがあることも考えられるが，ここでは安全のため，モジュールに変更があった場合には，すべてのファイルが再コンパイルされる設定としている．なお，モジュールファイルを使用しない場合には，

```
COMMON_MOD =
```

として，= のすぐ後で改行し，空のマクロとしておけばよい．

Makefile は，プログラムが複数のファイルから構成されているときに有効性を発揮するが，プログラムが単一のファイルである場合でも，これを利用するとオプション等を毎回入力しなくて済むので便利である．

5.2 make によるコンパイルの方法

ソースファイルが存在するディレクトリ内において，前項に示した Makefile を用意し，シェルウインドウで make と入力すると以下の表示が出力され，実行ファイル go が作られる（% はプロンプトを表し，入力は不要）．

```
% make
g95 -c interface.f90 -fimplicit-none -fbounds-check
g95 -c globals.f90 -fimplicit-none -fbounds-check
g95 -c main.f90 -fimplicit-none -fbounds-check
g95 -c sub1.f90 -fimplicit-none -fbounds-check
g95 -c sub2.f90 -fimplicit-none -fbounds-check
g95  -o go interface.o globals.o main.o sub1.o sub2.o
```

初めてコンパイルする場合には，上記のように各ファイルのオブジェクトファイルが作られ，最後にそれらから実行ファイル go が作られる．この後で，./go と入力すれば演算が実行される．

プログラムを編集し，sub1.f90 の内容が修正されたとする．これをコンパイルするには，再び make と入力する．すると，以下のように実行ファイルが作られる．

```
% make
g95 -c sub1.f90 -fimplicit-none -fbounds-check
g95  -o go interface.o globals.o main.o sub1.o sub2.o
```

上記では，sub1.f90 のみが新しくコンパイルされて，実行ファイル go が作られている．このように，make を用いると，不要なコンパイルを自動的に省略することができる．sub2.f90 あるいは main.f90 が修正された場合も同様で，該当するファイルのみが再コンパイルされる．一方，モジュール globals.f90 に変更があった場合には，make と入力すると，初めてコンパイルする場合と同様に，すべてのファイルが再コンパイルされる．

3.11 節で述べたように，モジュールの記述に変更があった場合には，そのモジュールを含むファイルだけでなく，モジュールを使用するすべてのプログラム単位を含むファイルを再コンパイルする必要がある．しかし，そのような依存関係の解析は実際には容易ではなく，依存関係の解析が不完全であるとプログラムは正常に動作しない．これを防ぐために，上記に示された Makefile では，すべてのファイルを再コンパイルすることとしている．

5.3 すべてのファイルを再コンパイルするスクリプト

4.6 節で示された「モジュール副プログラム型」の構成において，モジュール副プログラムを含む複数のファイルが存在する場合には，モジュールの依存関係が複雑になり，簡単な Makefile では対応できない可能性がある．この場合には，

- モジュールの依存関係を正確に解析して，その結果を基に Makefile を作成するツールを利用する．
- すべてのファイルを毎回再コンパイルする．

という方法が考えられる．前者の定番というべきツールはまだ存在しないようであるので，ここでは後者の方法を簡単に記しておく▼37．

複数のファイルをコンパイルして実行ファイルを作成するには，3.11.1 項で述べたように，以下のようにファイルを 1 つずつコンパイルしてもよい．

▶35　行頭にタブを入力することは，多くの場合に許可されているようである．

▶36　4.6 節の「混在型」の場合に，モジュール副プログラムを含むファイルが他にあれば，そのファイル名をここに書いておけばよい．

▶37　Windows PC では，p.226 に示したバッチファイルを作ればよい．

```
% g95 -c module.f90
% g95 -c prog.f90
% g95 module.o prog.o
```

ただし，この方法では入力が面倒であるので，上記のコマンドを書いたスクリプトを作成し，それをコンパイルのためのコマンドとして使用するという手段がある．

ソースファイルが存在するディレクトリ内において，次のような内容のテキストファイルを作成し，これを例えば compile という名称で保存する．

```
g95 -c params.f90
g95 -c subprogs.f90
g95 -c main.f90
g95 -o go params.o subprogs.o main.o
```

そして，次のように chmod コマンドを用いて，ファイルを実行可能な形式とする．

```
% chmod u+x compile
```

このようにすると，次回から

```
% ./compile
```

と入力するだけで，全てのファイルが再コンパイルされる．ファイルが追加された場合には，スクリプトファイル compile をエディタで開いて，これをコンパイルするための記述を追加する．

参考文献

[1] 牛島省：OpenMP による並列プログラミングと数値計算法，丸善株式会社，2006.

[2] JIS X 3001-1:1998（ISO/IEC 1539-1:1997）プログラム言語 Fortran——第 1 部：基底言語，日本規格協会，1998.

[3] SOJIN 編訳 C. K. Caldwell 編著：素数大百科，共立出版，2004.

[4] 森正武：数値解析，共立出版株式会社，2002.

[5] 一松信：数値解析，朝倉書店，2001.

[6] Cooper Redwine: *Upgrading to Fortran 90*, Springer, 1995.

[7] 藤野清次，張紹良：反復法の数理，朝倉書店，1996.

[8] R. S. バーガ（渋谷政昭訳）：計算機による大型行列の反復解法，サイエンス社，1975.

[9] 仁木滉，河野敏行：楽しい反復法，共立出版株式会社，1998.

[10] 戸川隼人：マトリクスの数値計算，オーム社，1971.

[11] 戸川隼人：共役勾配法，教育出版，1993.

[12] H. A. Van Der Vorst: "BI-CGSTAB: A first and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems." *SIAM J. Sci. Stat. Comput.*, Vol. 13, pp. 631–644, 1992.

[13] 谷内俊弥，西原功修：非線形波動，岩波書店，1998.

[14] 戸川隼人：計算機のための誤差解析の基礎，サイエンス社，1979.

[15] 大竹敢，矢吹道郎：使いこなす gnuplot，テクノプレス，2004.

[16] ロバートメクレンバーグ：*GNU Make*，オライリー・ジャパン，2005.

[17] アンドリュー・オラム，スティーブ・タルボット：make 改訂版，オライリー・ジャパン，1997.