

目次

目次.....	1
1. GCCとは	1
2. 作業フォルダの作成.....	1
3. 作業の概略	2
4. エディタを用いたプログラムの作成	3
4. 1 mi の起動 (①)	3
4. 2 プログラムの作成 (②)	3
4. 3 プログラムの保存 (③)	4
5. GCCを用いたコンパイル・リンク	5
5. 1 ターミナルの起動 (④)	5
5. 2 カレントフォルダの移動 (⑤) とファイルの表示.....	6
5. 3 gcc を用いてコンパイル (⑥)	7
5. 4 コンパイル・リンク時のエラー (⑦)	7
6. 実行 (⑧)	8

1. GCCとは

GCC(ジーシーシー)は、GNU の GNU Compiler Collection (グニューコンパイラコレクション) が開発したコンパイラ群のことです。フリーソフトウェアとして公開されており、Linux や FreeBSD の開発に不可欠のものとなっています。また、現在では多くの環境で標準的な C コンパイラとして採用されています。

このテキストでは、**GCC**の C 言語用コンパイラである"**gcc**"のプログラムを用いて、作成したプログラムをコンパイル、リンクし、実行ファイルを作成する手順について説明します。

Tips: 参考

GNU コンパイラコレクション:

<https://ja.wikipedia.org/wiki/GNUコンパイラコレクション> (Wikipedia, 参照日2023年9月20日)

2. 作業フォルダの作成

プログラミングによって、ソースプログラムと実行ファイルが作成され、頻繁に書き換えや消去が行われます。他の文書ファイルなどと混在して不注意でファイルを消さないためにも、予めソースプログラムや実行ファイルを保存する**作業フォルダ**を作成しておきます。

Mac コンピュータでは、書類/Documents)フォルダ下にプログラミング用の作業フォルダを作成してください。

Finder から書類/Documents)を開きます。次に、メニューバーにある[ファイル]メニューを開き、[新規フォルダ]を選びます(図 1)。フォルダの名前はわかりやすいものが良いので、ここでは、"**GC**"とつけることにします(図 2)。この授業では常にこのフォルダ内で作業を行うことにすれば、他のファイルと混同することがありません。



図1 書類へのフォルダの作成

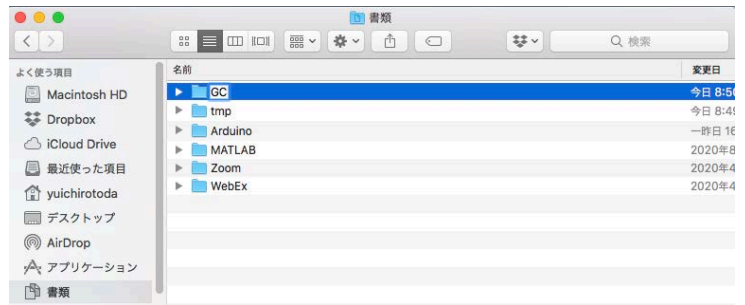


図2 GC フォルダの作成

3. 作業の概略

プログラムの作成は、大きく分けて

- ・ エディタを用いたプログラムの作成 (第4節)
- ・ ターミナルでの gcc によるコンパイル (第5節)
- ・ プログラムの実行 (第6節)

に分けられます。これらの作業を行っているときのデスクトップの様子が図3です。右側のウィンドウで `mi` を用いたプログラムの入力、左側のウィンドウでコマンドプロンプトによる作業をそれぞれ行っています。以降の4節～6節では、これらの作業について説明します。

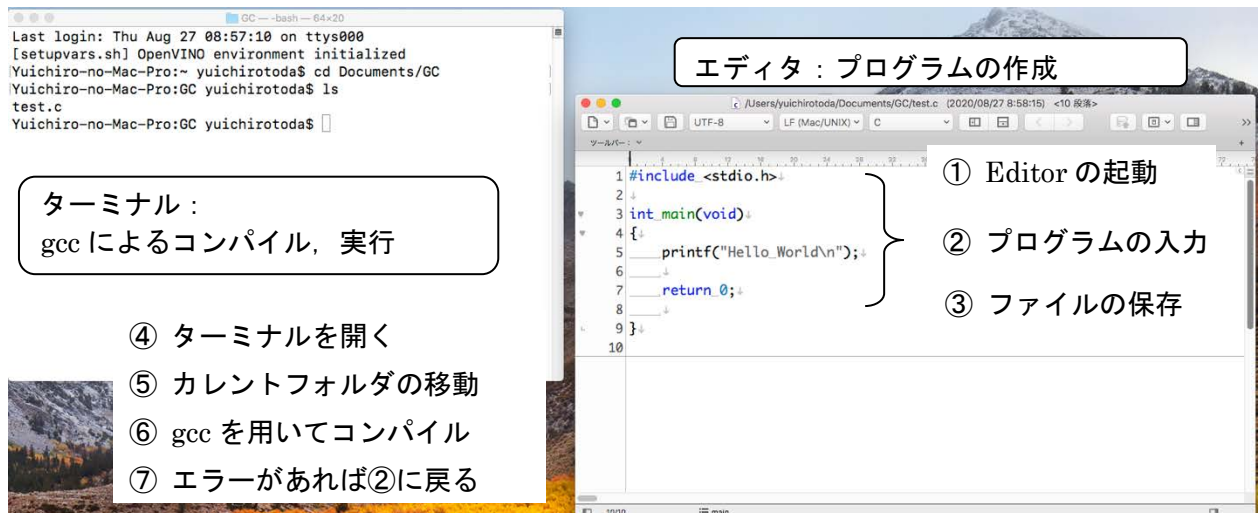


図3 プログラム作成, コンパイル, 実行時のデスクトップの様子

4. エディタを用いたプログラムの作成

プログラムを入力するソースファイルの形式は、テキスト形式です。従って Mac に付属のテキストエディタでも入力できます。ここでは、便利なフリーソフトウェア **mi** を用いたプログラムの入力方法を説明します。

4. 1 mi の起動 (①)

mi は **mi** のサイト (<https://www.mimikaki.net/>) からダウンロードしてインストールを行うことで使用可能になります。アプリケーションを起動する際は、下記の 2 種類のいずれかの方法で起動するのが簡単です。

- ・ [Finder]-[アプリケーション]-[mi]を選んでダブルクリック。(図 4)
- ・ [Launchpad]を開き[mi]をクリック。(図 5)

すると **mi** の画面が表示され、新たなソースファイルを作成できる状態になります。



図 4 アプリケーションから **mi** を起動

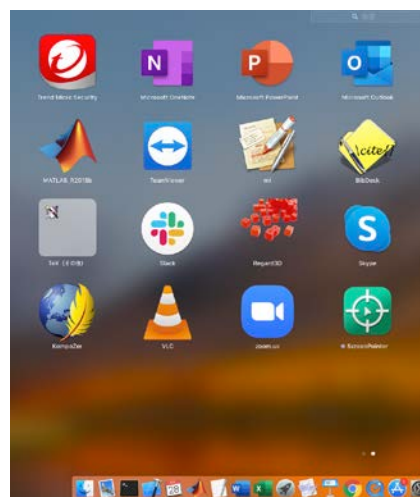


図 5 Launchpadから**mi** をクリック

4. 2 プログラムの作成 (②)

プログラムを入力していきます。

サンプルファイルとして、図 6 のプログラムを入力してください。プログラムは全て半角で入力してください。全角文字を入力すると、コンパイル時にエラーが表示されます。(例外として、コメント部分(`/* */`で囲まれる部分(テキスト p.25)と、書式付入出力関数(テキスト p.112)のダブルコーテーション(`"`)で囲まれる範囲には、全角文字での入力が許されています)また、`printf`や`return` の左側の空白 (インデント) の入力には `TAB` キーを使います。さらに、**mi** では、入力内容に応じて自動的にインデントが行われます。 `{ }` の位置を同じ列にするとプログラムの構造が分かりやすいものとなります。

```
#include <stdio.h>

int main(void)
{
    printf("Hello¥n");

    return 0;
}
[EOF]
```

図 6 サンプルプログラム

4. 3 プログラムの保存 (③)

作成したプログラムを保存します。保存は[ファイル]-[別名で保存]を選びます。保存場所は、2.で作成した書類の GC フォルダになるようにしてください。(図 7)

ファイル名の命名には、次の点を守ってください。

- ・ ファイル名はできるだけ半角文字で入力してください。全角文字でもコンパイルは可能ですが、作業時の入力の手間が増えます。
- ・ ファイルの名前の末尾に、".c"を半角文字で必ずつけてください。これは作成したファイルが C 言語のファイルであることを示す拡張子の規則であり、これ以外の拡張子をつけると、誤動作の原因となります。(ファイル名の例 : test.c)

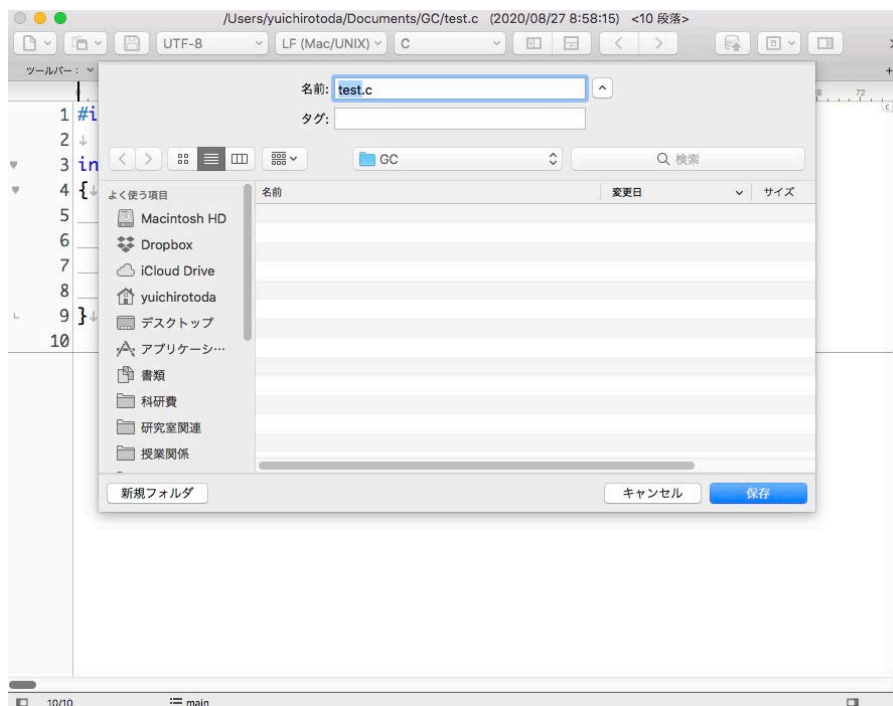


図 7 mi でのファイルの保存

正しく保存されると、mi のプログラム入力画面の各文字や記号の色が変わります。これは入力したファイルを mi が C 言語のファイルと認識したためで、C 言語の予約語や関数、波括弧({})などに色が付きます。これによってプログラムの構造がわかりやすくなるだけでなく、入力ミスなどを見つける手がかりにもなります。既に作成した C 言語のソースファイルを開いたときは、初めから該当する文字・記号に色が付きます。

コンパイル・リンク時にエラー表示が出た場合は、再度 mi でプログラムを修正します。このため mi のウィンドウはプログラムが完成するまでは開いたままの方がよいでしょう。

Tips: 他メディアへの保存 (バックアップ)

作業フォルダ内には、ソースファイル(拡張子:".c")と実行ファイル(拡張子:".exe")が存在します。大切なのはソースファイル(".c")の方です。Z ドライブの内容が消える場合もありますので、重要なソースファイルは USB メモリ等にバックアップ (コピー) しておくとい良いでしょう。

5. GCCを用いたコンパイル・リンク

プログラムが入力できたら、次にプログラムをコンパイルします。コンパイルは、ソースプログラムに書かれた人間に分かりやすい命令文を、マシン語に翻訳する作業で、この作業を行わないと、コンピュータは命令文を理解することができません。さらにリンクは、作成したプログラムをあらかじめ用意されたライブラリ関数と結びつけ、実行可能なプログラムを作成する作業です。これらの作業が問題なく終了して初めて、コンピュータで実行可能なプログラムが完成します。

GCCでは、gcc プログラムでコンパイルとリンクを一度に行えます。

5. 1 ターミナルの起動 (④)

gcc はターミナル(コマンドウィンドウ、コンソール画面とも呼ばれる)で直接プログラム名を入力して起動させるプログラムです。ターミナルは、UNIX コマンドを入力するためのエントリーポイントを提供する Mac の機能です。特徴は、Mac の GUI (Graphical User Interface)を使用せずにコンピュータでタスクを実行できる点です。

gcc を使用するためには、まずターミナルを起動します。コマンドプロンプトは、Finder から「アプリケーション」→「ユーティリティ」を開くとアプリケーションがありますのでダブルクリックで開いてください。(図8)

ターミナルを起動すると、図9のような画面が開きます。“\$”の右側で点灯する灰色の四角をカーソルと呼び、キーボードから入力した文字がカーソルの位置に表示されます。文字列を入力して Enter キーを押すと、指定するプログラムが実行されます。(Windows より前のコンピュータは、全てコマンドプロンプトで制御していました。また Linux でも基本的に同様の方法で制御します。)

“\$”の左側の文字は、“PC 名:カレントフォルダ(現在作業を行っているフォルダ) ユーザ名”を示しています。例えば図9では、

Yuichiro-no-Mac-Pro:~ yuichiro toda\$

と表示されています。これは、カレントフォルダが“~”であることを示しています。(“~”はホームディレクトリを意味し、Users フォルダがカレントフォルダになっていることを意味しています。)

ターミナルのウィンドウは、左上の×をクリックすれば、閉じます。



図8 ターミナルの起動方法

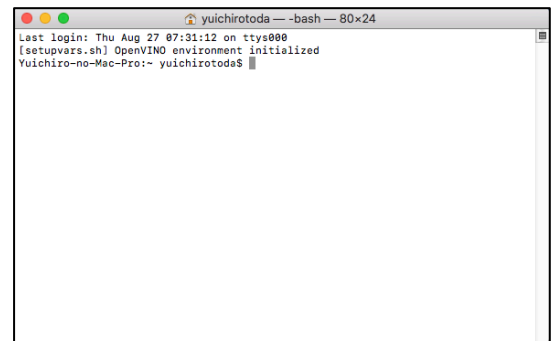


図9 ターミナル

Tips: 参考

ターミナルによる Mac 操作。13 の基本コマンド一覧と使い方

: <https://original-game.com/mac-terminal-command/> (参照日 2023 年 9 月 20 日)

5. 2 カレントフォルダの移動 (⑤) とファイルの表示

コマンドプロンプトを起動した時のカレントフォルダは、ログインしたユーザの個人設定を保存するホームディレクトリの"~"となっています。これを 2. で作成した"Documents"の"GC"に移動します。

移動には、カレントフォルダを移動する命令(cd)を用います。ターミナルでは、大文字／小文字の区別はありません。

・ cd	カレントフォルダを移動する命令
cd	ホームディレクトリ(“~”)に移動
cd ..	カレントフォルダを一つ上のフォルダに移動
cd /	もっとも上位のフォルダに移動

これらのコマンドを用いて、 "Documents"の"GC"に移動する方法を、下記に示します。

方法 1 :

\$cd "My Documents"	←"Documents"に移動
\$cd GC	←"Documents"の下の GC に移動

方法 2 :

\$cd Documents/GC	←" Documents"の下の GC に移動
-------------------	-------------------------

Tips: 文字の補完

cd my まで入力し、TAB キーを押すと残りの文字が補完されて、cd "My Documents"と簡単に入力できます。

カレントフォルダ内のフォルダやサブフォルダの情報を表示するのが、ls というコマンドです。

・ ls	ファイル・フォルダのリストなどを表示する命令
-------------	------------------------

ls コマンドを用いて、カレントフォルダに 4. で作成したプログラムが正しく保存されていることを確認してください。(図 10)

5. 3 gcc を用いてコンパイル (⑥)

最も簡単なコンパイル方法は、

gcc ファイル名

です。しかしこのままでは作成されるプログラム名が"a.out"になってしまいますので、下記のように入力するようにしてください。

・プログラムをコンパイル・リンクするコマンド

gcc **ファイル名** -o **実行ファイル名** -lm -ansi -pedantic -Wall

-o 実行ファイル名を指定するオプション。その後のファイル名が実行ファイル名となる。実行ファイルの拡張子は、必ず".exe"とする。

-lm 数学関数(sin, sqrt 等)を用いる場合に必要なオプション。数学関数を使っていたら、それに必要なライブラリがリンクされる。

-ansi -pedantic 厳格な ANSI 仕様でのコンパイル。(本講義必須)

-Wall 文法チェックで疑わしい箇所を全て表示する。(推奨)

例 : gcc **test.c** -o **test.exe** -lm -ansi -pedantic -Wall

ソースプログラムが **test.c**, 実行ファイル名が **test.exe**

Tips: gcc のオプション

gcc のオプションについては、

gcc --help

と入力することによって見るすることができます。必要に応じて参考にしてください。

Tips: コマンドプロンプトの履歴機能

コマンドプロンプトでは同じ様な入力をする場合、矢印キーの"↑"を押すと前に入力したコマンド内容呼び出せます。この履歴はいくらでも過去にさかのぼることができ、また呼び出したものを部分的に修正して使うことも可能です。

5. 4 コンパイル・リンク時のエラー (⑦)

エラーがなければ、実行ファイル(test.exe)がカレントフォルダに作成されます。ls コマンドを用いて、実行ファイルの作成日時が最新であることを確認してください。(図 10)

入力ミスやプログラムの文法的なエラーがある場合は、gcc がコンパイル・リンク時にチェックしてエラーを表示し、コンパイル・リンクの作業を中止します。エラーが表示される場合は、エラーの内容を参考に、プログラムを修正してください。

エラーの例を 3 つあげておきます。ソースファイル名(test.c)の後の数字がエラーと考えられるプログラムの先頭からの行数を示しています。その行かその前にエラーがある場合が多いです。

例 1 : 文末にセミコロン(;)が無い場合

```
Yuichiro-no-Mac-Pro:GC yuichiro toda$ gcc test.c -o test.exe -lm -ansi -pedantic -Wall
test.c:5:19: error: expected ';' after expression
    printf("Hello\n")
                   ^
                   ;
1 error generated.
```

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

例 2 : ダブルコーテーション(")が不足する場合

```
Yuichiro-no-Mac-Pro:GC yuichirotda$ gcc test.c -o test.exe -lm -ansi -pedantic -Wall
test.c:5:9: warning: missing terminating ' character [-Winvalid-pp-token]
    printf("Hello¥n");
    ^
test.c:5:9: error: expected expression
test.c:10:1: error: expected '}'
^
test.c:4:1: note: to match this '{'
{
^
1 warning and 2 errors generated.
```

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

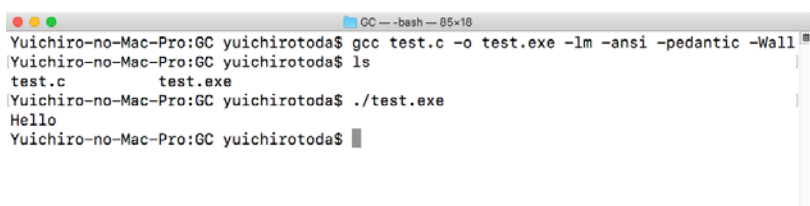
例 3 : 関数名を間違えている場合

```
Yuichiro-no-Mac-Pro:GC yuichirotda$ gcc test.c -o test.exe -lm -ansi -pedantic -Wall
test.c:5:2: warning: implicit declaration of function 'print'
[-Wimplicit-function-declaration]
    print("Hello¥n");
    ^
1 warning generated.
Undefined symbols for architecture x86_64:
  "_print", referenced from:
      _main in test-f263a9.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

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

6. 実行 (⑧)

正しくプログラムがコンパイルできれば、ターミナルで”./実行ファイル名(ドット, スラッシュ, ファイル名)”を入力して Enter キーを押して下さい。プログラムが実行されます。(図 1 0)



```
Yuichiro-no-Mac-Pro:GC yuichirotda$ gcc test.c -o test.exe -lm -ansi -pedantic -Wall
Yuichiro-no-Mac-Pro:GC yuichirotda$ ls
test.c      test.exe
Yuichiro-no-Mac-Pro:GC yuichirotda$ ./test.exe
Hello
Yuichiro-no-Mac-Pro:GC yuichirotda$
```

図 1 0 実行ファイルの確認と、プログラムの実行

プログラムが予定通り実行されない場合は、再度プログラムを修正し、コンパイルしなおして下さい。(②へ戻る)

Tips: プログラムが終了しない or 動き続けて止まらない

実行したプログラムにバグがあって終了できなくなってしまうたら、たいていの場合は **Ctrl + C** で強制終了できます。C プログラムが無限ループに陥った場合も、この操作でプログラムを終了させることができます。

Tips: それでも止まらない (ハングアップした) 場合の対処

ハングアップしたときにコンピュータの電源を切ることは、ハードディスクなどに物理的損傷を与えたり Mac システムファイルを壊したりする場合があります。下記の操作でアプリケーションの強制終了を起動して、該当プログラムを強制終了してください。

Option + Command + ESC (Option キー, Command キー, ESC キーを同時に押す)