数値解析特論I

- Unixコンソールコマンドの復習
- C言語によるプログラミング—初歩

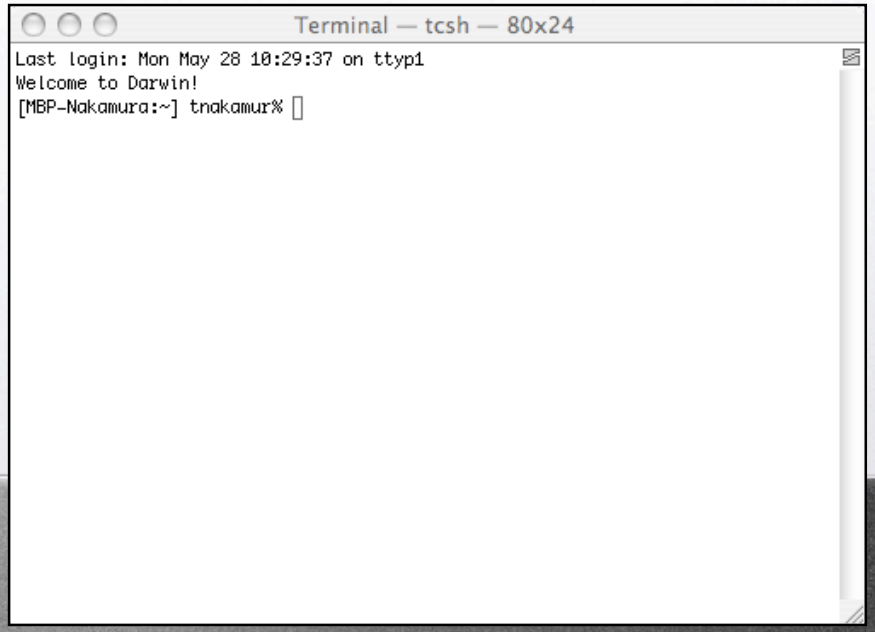
Unix系OSのコンソールコマンド復習

- コンソールから様々なコマンドを入力して操作を行う。

- コンソール（ターミナル）の起動

  Macintosh HD →アプリケーション→

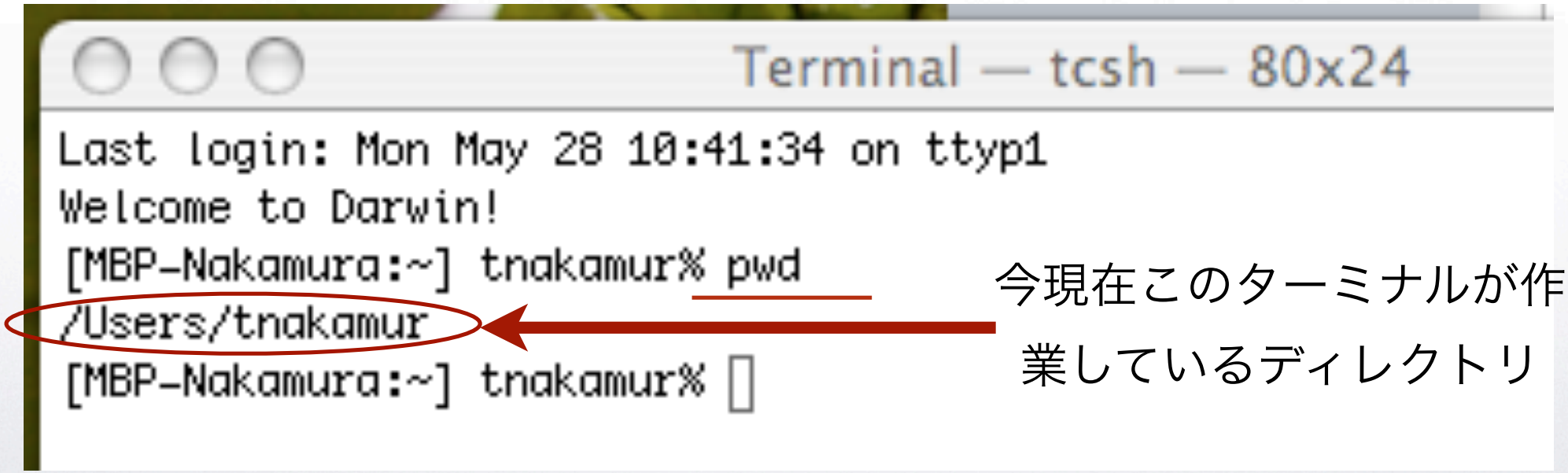
  ユーティリティー　→ターミナル　をダブルクリック



Unix系OSのコンソールコマンド復習

- 現在作業しているディレクトリを調べる

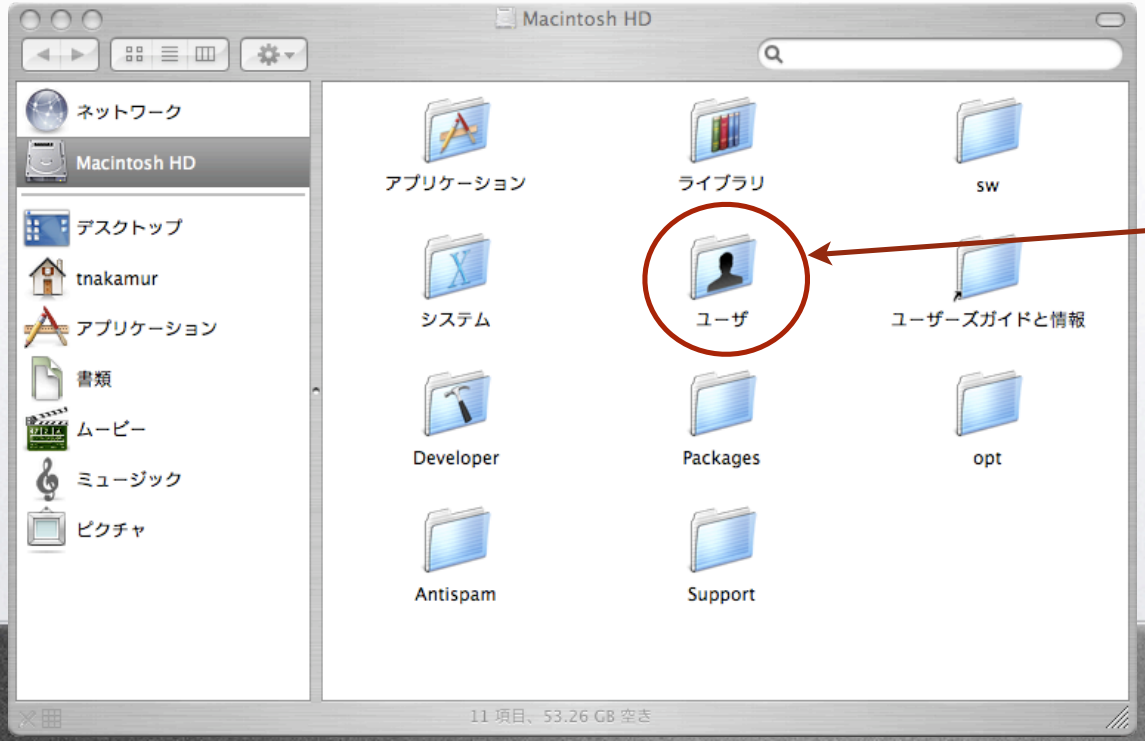
% pwd ↓



今現在このターミナルが作業しているディレクトリ

確認してみる　“/Users/tnakamur”

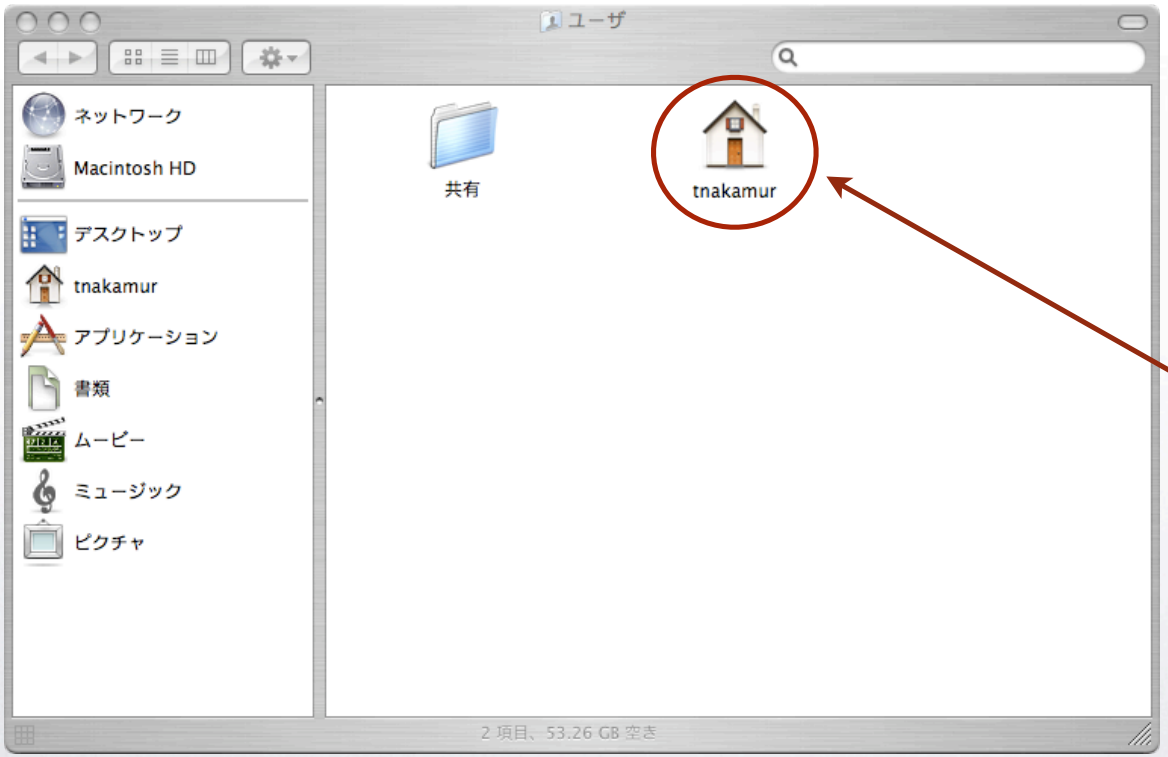
- /Users/tnakamur → “Users”というディレクトリの中にある
  ”tnakamur”というディレクトリという意味
- “Macintosh HD”をダブルクリック



Users(ユーザ)というディレクトリ

確認してみる　“/Users/tnakamur”

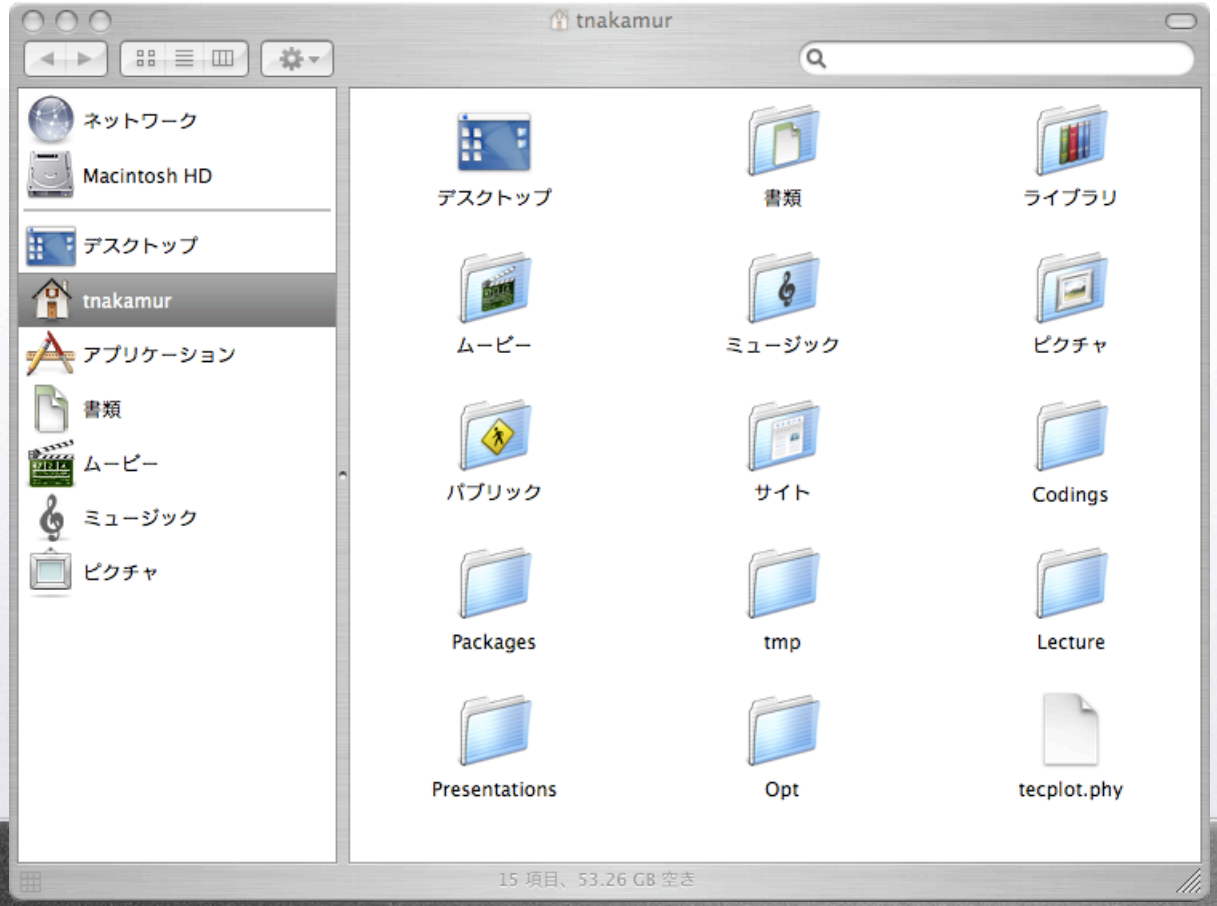
- “ユーザ(Users)”をダブルクリック



tnakamurという
ディレクトリ

確認してみる　“/Users/tnakamur”

- “tnakamur”をダブルクリック



ターミナルは今現在ここで作業している。

Unix系OSのコンソールコマンド復習

- 現在作業しているディレクトリに含まれるファイル・ディレクトリの一覧を表示する(list)

% ls ↓

Terminal — tcsh — 80x24

```
Last login: Mon May 28 10:41:34 on ttyp1
Welcome to Darwin!
[MBP-Nakamura:~] tnakamur% pwd
/Users/tnakamur
[MBP-Nakamura:~] tnakamur% ls
Codings/        Lecture/        Music/          Pictures/       Sites/
Desktop/        Library/        Opt/            Presentations/  tecplot.phy
Documents/      Movies/         Packages/       Public/         tmp/
[MBP-Nakamura:~] tnakamur%
```

先ほどの内容と比較してみる

Unix系OSのコンソールコマンド復習

- ディレクトリを作成

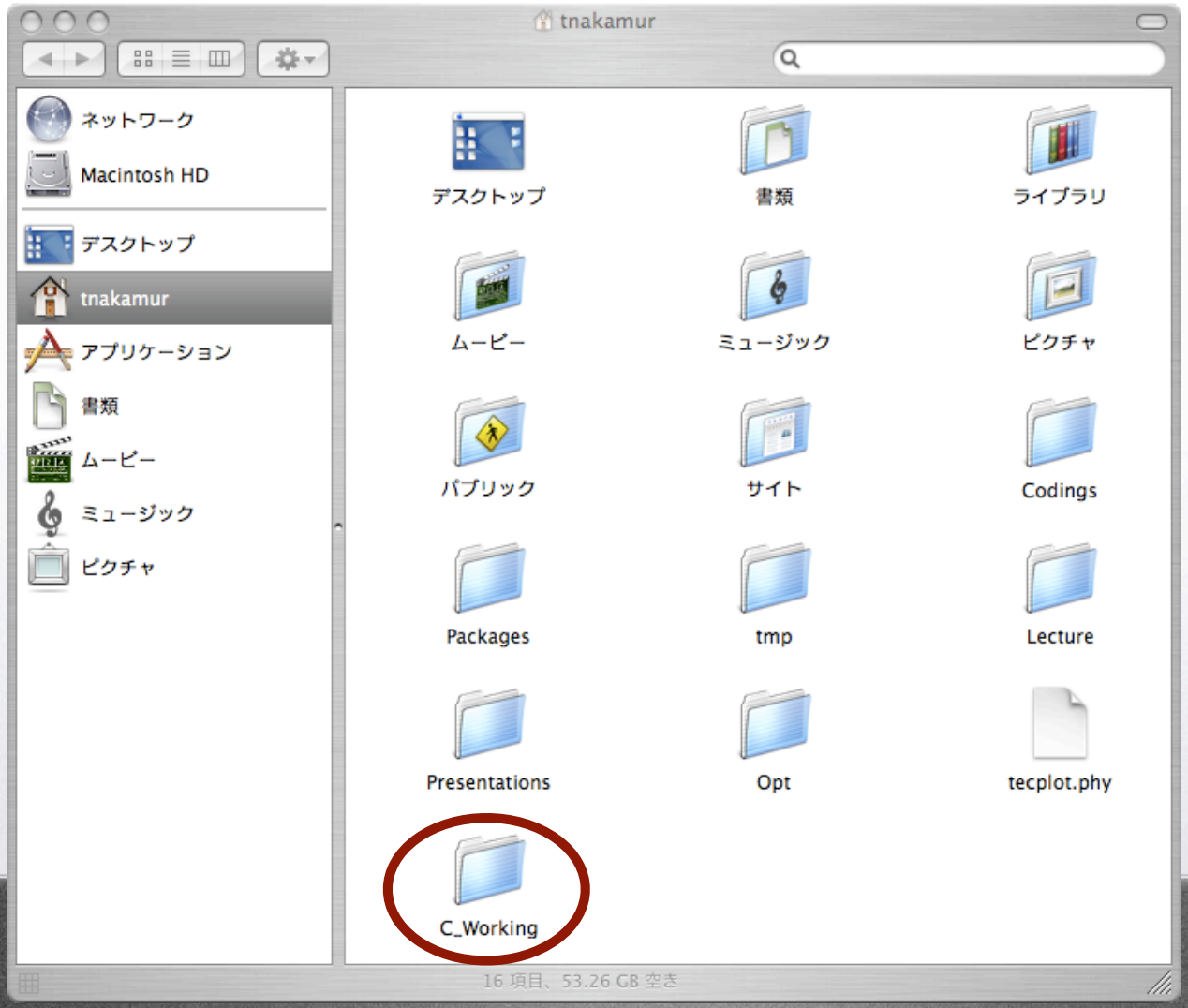
  % mkdir <作成するディレクトリ名>↓

Terminal — tcsh — 80x24

```
Last login: Mon May 28 10:41:34 on ttyp1
Welcome to Darwin!
[MBP-Nakamura:~] tnakamur% pwd
/Users/tnakamur
[MBP-Nakamura:~] tnakamur% ls
Codings/        Lecture/        Music/          Pictures/       Sites/
Desktop/        Library/        Opt/            Presentations/  tecplot.phy
Documents/      Movies/         Packages/       Public/         tmp/
[MBP-Nakamura:~] tnakamur% mkdir C_Working
[MBP-Nakamura:~] tnakamur% ls
C_Working/      Lecture/        Opt/            Public/
Codings/        Library/        Packages/       Sites/
Desktop/        Movies/         Pictures/       tecplot.phy
Documents/      Music/          Presentations/  tmp/
[MBP-Nakamura:~] tnakamur%
```

確認してみる　“/Users/tnakamur”

- C_Workingが増えている



Unix系OSのコンソールコマンド復習

- ディレクトリを移動する

  % cd <移動する先のディレクトリ名>↓

Terminal — bash — 80x24

```
Last login: Mon May 28 10:45:08 on ttyp1
Welcome to Darwin!
[MBP-Nakamura:~] tnakamur% bash
MBP-Nakamura:~ tnakamur$ pwd
/Users/tnakamur
MBP-Nakamura:~ tnakamur$ ls
C_Working       Lecture         Opt             Public
Codings         Movies          Packages        Sites
Desktop                         Pictures        tecplot.phy
Documents       Music           Presentations   tmp
MBP-Nakamura:~ tnakamur$ cd C_Working
MBP-Nakamura:~/C_Working tnakamur$ pwd
/Users/tnakamur/C_Working
MBP-Nakamura:~/C_Working tnakamur$ ls
MBP-Nakamura:~/C_Working tnakamur$
```

移動前の場所

ここに移動する

移動

移動後（今現在）の場所

Unix系OSのコンソールコマンド復習

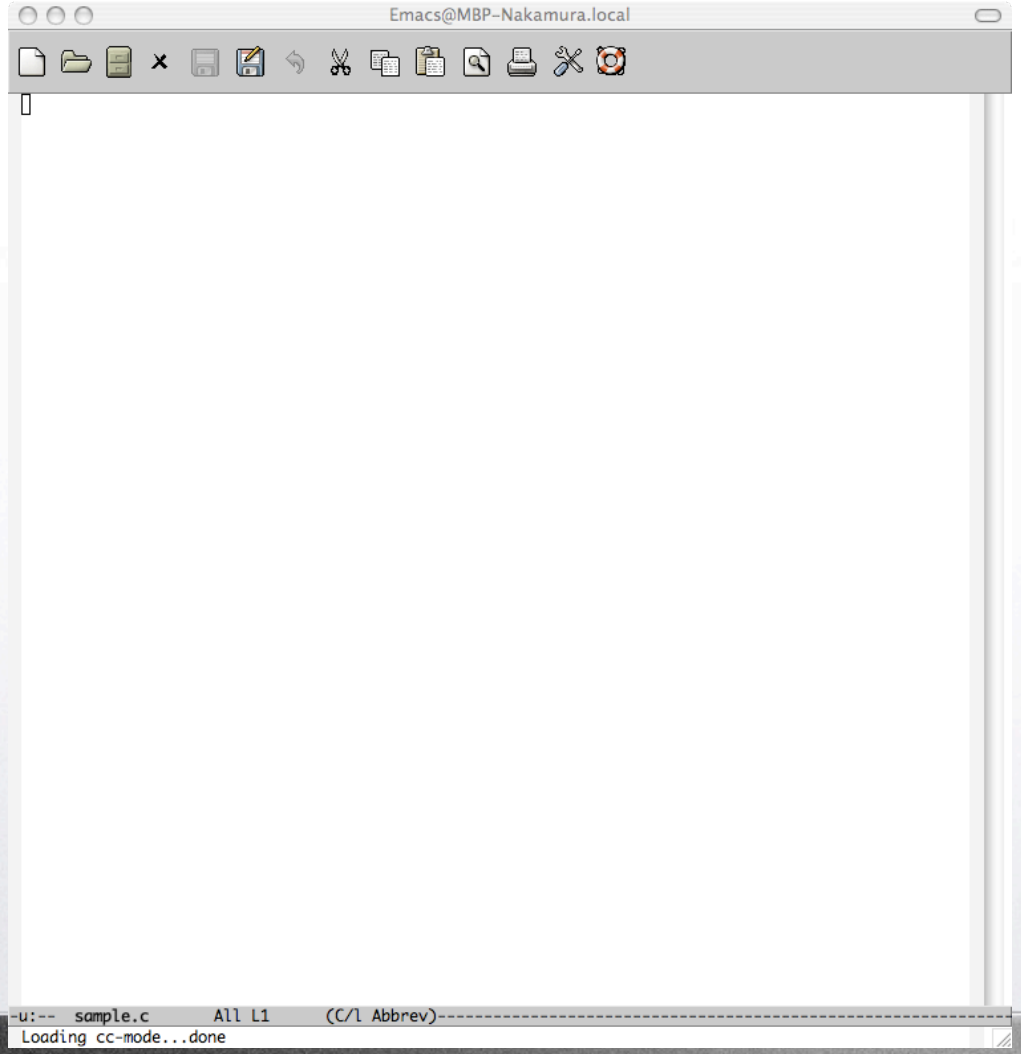
- テキストファイルを作成・編集（エディタの利用）

% emacs <作成・編集するファイル名> &↓

Terminal — bash — 80x24

```
Last login: Mon May 28 10:45:08 on ttyp1
Welcome to Darwin!
[MBP-Nakamura:~] tnakamur% bash
MBP-Nakamura:~ tnakamur$ pwd
/Users/tnakamur
MBP-Nakamura:~ tnakamur$ ls
C_Working       Lecture         Opt             Public
Codings         Library         Packages        Sites
Desktop         Movies          Pictures        tecplot.phy
Documents       Music           Presentations   tmp
MBP-Nakamura:~ tnakamur$ cd C_Working
MBP-Nakamura:~/C_Working tnakamur$ pwd
/Users/tnakamur/C_Working
MBP-Nakamura:~/C_Working tnakamur$ ls
MBP-Nakamura:~/C_Working tnakamur$ emacs sample.c&
```

Emacs エディタの起動



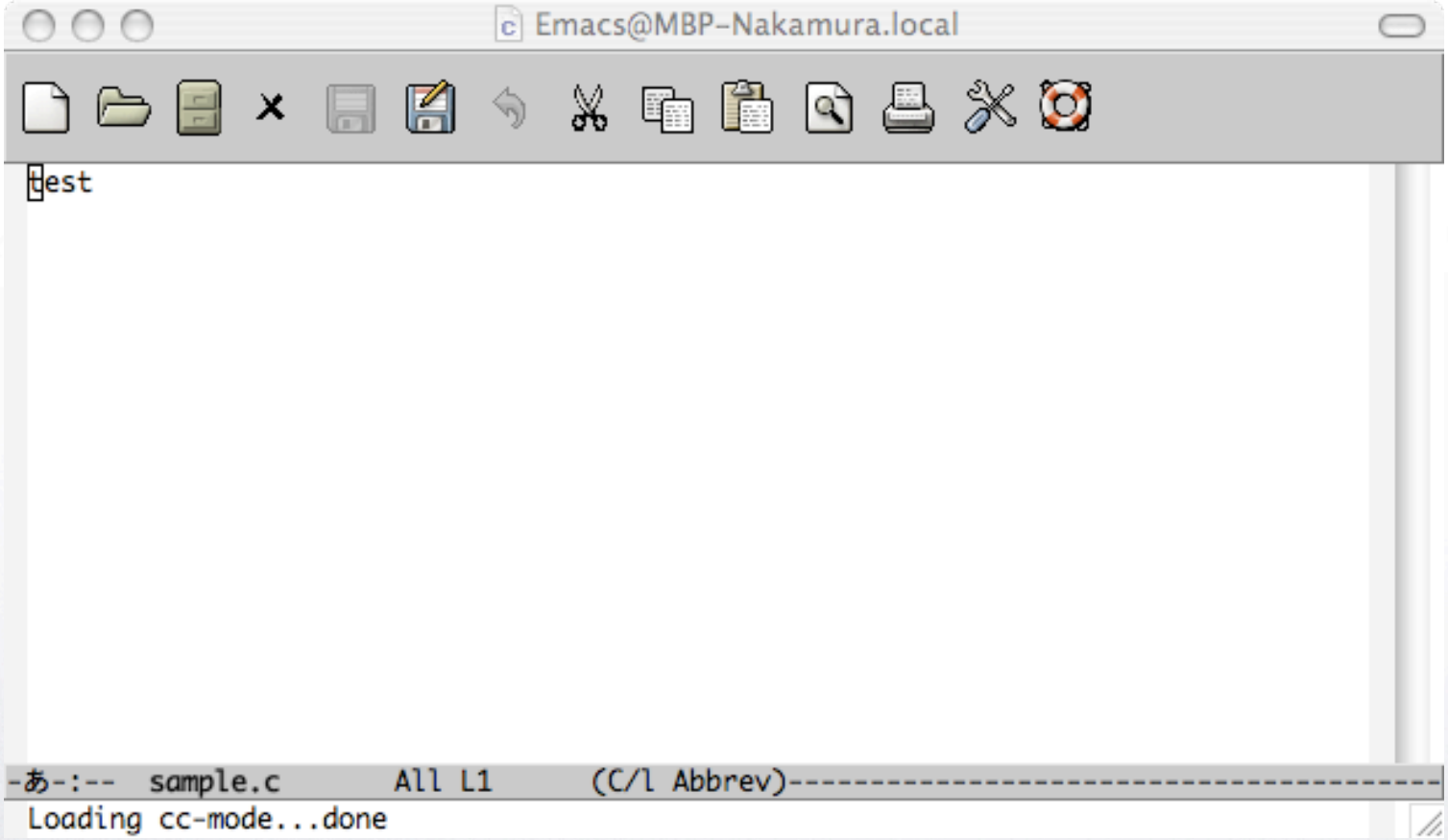
若干見た目が異なる

かも？

Emacs エディタの各種コマンド

- 保存：Ctrl を押しながら, XとSを順番に押す。
- 終了：Ctrlを押しながら, XとCを順番に押す。
- カット：Ctrlを押しながら,Kを押す。
- ペースト：Ctrlを押しながら,Yを押す。
- やり直し：Ctrlを押しながら,XとUを順番に押す。
- 文字を検索：Ctrlを押しながら、Sを押す。
- 文字を置換：Escを押して、%を押す。
- その他いっぱいあります。

Emacs エディタの起動



適当な文字を入力して、Ctrl +X,Cで保存して終了してみる。

確認してみる　“/Users/tnakamur”

- ターミナルでls

```
MBP/Users/tnakamur/C_Working 10> ls
sample.c
MBP/Users/tnakamur/C_Working 11>
```

- ファイルの中身を表示させるコマンド “less”

% less <ファイル名>↓

```
MBP/Users/tnakamur/C_Working 15> ls
sample.c
MBP/Users/tnakamur/C_Working 16> less sample.c
```

ファイル”sample.c”をコピーする。

- ターミナルでcp

% cp <コピー元のファイル名> <コピー先のファイル名> ↓

```
MBP/Users/tnakamur/C_Working 15> ls
sample.c
MBP/Users/tnakamur/C_Working 16> less sample.c
MBP/Users/tnakamur/C_Working 17> cp sample.c sample_1.c
MBP/Users/tnakamur/C_Working 18> ls
sample.c        sample_1.c
MBP/Users/tnakamur/C_Working 19>
```

ファイル”sample.c”をコピーする。

- sample_2.c sample_3.c sample_4.c ,,,にもコピーしてみる。

```
MBP/Users/tnakamur/C_Working 17> cp sample.c sample_1.c
MBP/Users/tnakamur/C_Working 18> ls
sample.c        sample_1.c
MBP/Users/tnakamur/C_Working 19> cp sample.c sample_2.c
MBP/Users/tnakamur/C_Working 20> cp sample.c sample_3.c
MBP/Users/tnakamur/C_Working 21> cp sample.c sample_4.c
MBP/Users/tnakamur/C_Working 22> cp sample.c sample_5.c
MBP/Users/tnakamur/C_Working 23> ls
sample.c        sample_2.c      sample_4.c
sample_1.c      sample_3.c      sample_5.c
MBP/Users/tnakamur/C_Working 24>
```

ファイル名”sample.c”の変更

- ファイルの移動・名前の変更 mv

  % mv <元のファイル名> <新しいファイル名> ↓

```
MBP/Users/tnakamur/C_Working 23> ls
sample.c        sample_2.c      sample_4.c
sample_1.c      sample_3.c      sample_5.c
MBP/Users/tnakamur/C_Working 24> mv sample.c  new_sample.c
MBP/Users/tnakamur/C_Working 25> ls
new_sample.c    sample_2.c      sample_4.c
sample_1.c      sample_3.c      sample_5.c
MBP/Users/tnakamur/C_Working 26>
```

sample.c の名前が new_sample.cに変更

ディレクトリの表現のお約束

/Users/tnakamur/C_Working/working00
に取っての親ディレクトリ

/Users/tnakamur/C_Working　→　“..” で表す

```
MBP/Users/tnakamur/C_Working/working00 31> pwd
/Users/tnakamur/C_Working/working00
MBP/Users/tnakamur/C_Working/working00 32> cd ..
MBP/Users/tnakamur/C_Working 33> pwd
/Users/tnakamur/C_Working
MBP/Users/tnakamur/C_Working 34>
```

親ディレクトリに移動

正規表現（複数のファイルの指定）

/Users/tnakamur/C_Workingにあるsample_1.c sample_2.c, sample_3.c, sample_4.c, ...を全てworking00にコピーしてみる。

% cp sample_*.c working/

```
MBP/Users/tnakamur/C_Working 33> pwd
/Users/tnakamur/C_Working
MBP/Users/tnakamur/C_Working 34> ls
sample_1.c      sample_3.c      sample_5.c
sample_2.c      sample_4.c      working00/
MBP/Users/tnakamur/C_Working 35> cp sample_*.c working00/
MBP/Users/tnakamur/C_Working 36> cd working00/
MBP/Users/tnakamur/C_Working/working00 37> ls
sample_1.c      sample_3.c      sample_5.c
sample_2.c      sample_4.c      tmp_sample.c
MBP/Users/tnakamur/C_Working/working00 38>
```

sample_”何とか”.cが全てコピーされる

ファイルの消去

ファイル・ディレクトリの消去rm

% rm -rf <ファイル(ディレクトリ)名>

```
MBP/Users/tnakamur/C_Working/working00 37> ls
sample_1.c      sample_3.c      sample_5.c
sample_2.c      sample_4.c      tmp_sample.c
MBP/Users/tnakamur/C_Working/working00 38> rm -rf *
MBP/Users/tnakamur/C_Working/working00 39> ls
MBP/Users/tnakamur/C_Working/working00 40>
```

全て消去

C言語のプログラミング

- とにかくやってみましょう。

<u>プログラミングの手順</u>

1.計算方法などを考える。

2.プログラムのソースファイル(*.c)を書く

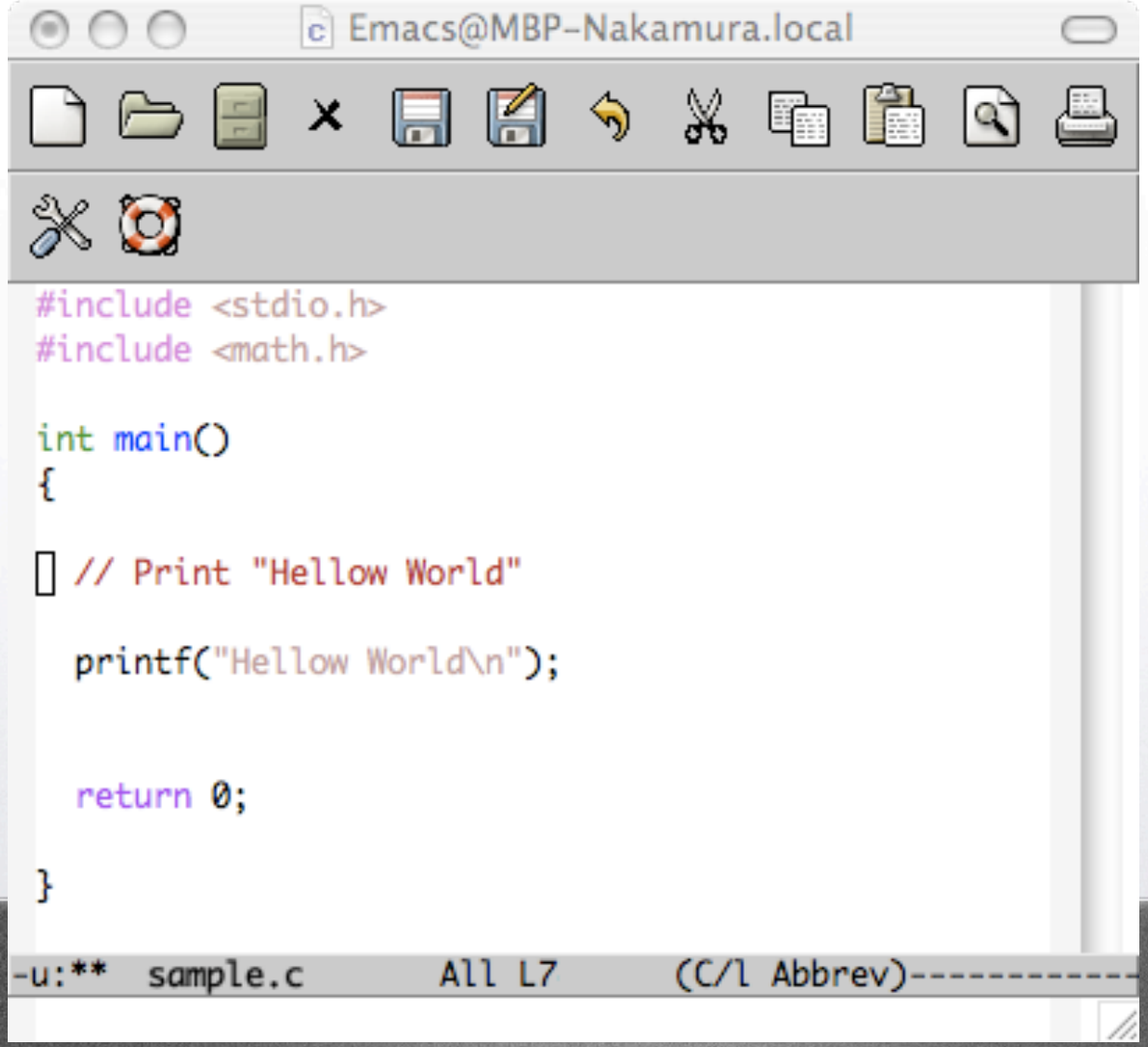
3.コンピュータの理解できる形式（実行形式）にソー

ファイルを変換する（コンパイル）

4.実行形式のプログラムを実行する

単純なプログラム

- emacs でsample.cを編集　%emacs sample.c & ↓



Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{

  // Print "Hellow World"

  printf("Hellow World\n");

  return 0;

}
```

おまじない！

ここからプログラム本体

“//”で始まる行は無視される

これは実行される。

行の最後は”;”を書く

おまじない！プログラムが終了する前に書く

ここまでプログラム本体

-u:** sample.c All L7 (C/l Abbrev)----------

- sample.cから実行形式のファイル”run”を作成（コンパイル）

% cc -o run sample.c -lm

cc:コンパイルコマンド
-o run :オプション(runという名前の実行形式を作成)
sample.c:変換するソースファイル

-lm : おまじない

```
MBP/Users/tnakamur/C_Working 66> ls
sample.c
MBP/Users/tnakamur/C_Working 67> cc -o run sample.c -lm
MBP/Users/tnakamur/C_Working 68> ls
run*            sample.c
MBP/Users/tnakamur/C_Working 69>
```

- runを実行

% ./run ↓

./:おまじない

```
MBP/Users/tnakamur/C_Working 69> ./run
Hellow World
MBP/Users/tnakamur/C_Working 70>
```

“Hellow World(改行)”と出力

printf 関数ー画面への出力

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{

  // Print "Hellow World"

  printf("Hellow World\n");


  return 0;

}
```

-u:** sample.c All L7 (C/l Abbrev)----------

printf(“Hellow World\n”);

\n :改行させる記号

単純な計算プログラム

- emacs でsample.cを編集　%emacs sample.c & ↓

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{
  int i,j;

  i=1;
  j=2;

  printf("I is %d J is %d\n",i,j);


  return 0;
```

変数の宣言

変数iに１を代入

変数jに２を代入

```
-u:--  sample.c        Top L11     (C/l Abbrev)----------
Wrote /Users/tnakamur/C_Working/sample.c
```

変数が整数

...,-2,-1,0,1,2,3,...
の場合

int型

- コンパイルして実行

% cc -o run sample.c -lm
%./run

```
MBP/Users/tnakamur/C_Working 70> cc -o run sample.c -lm
MBP/Users/tnakamur/C_Working 71> ./run
I is 1 J is 2
MBP/Users/tnakamur/C_Working 72>
```

iの値　jの値

printf 関数ー画面への出力

printf(“I is %d J is %d \n”,i,j); → “I is 1 J is 2” と出力

%d : 整数型の変数の値を出力

printf(“I is %d J is %d \n”, i, j);

この順番で%の場所に

を代入して出力する。

単純な計算プログラム

- emacs でsample.cを編集　　%emacs sample.c & ↓

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{
  int i,j,k;

  i=1;
  j=2;

  k=i+j;

  printf("I is %d J is %d K is %d\n",i,j,k);


  return 0;

}
```

```
-u:--  sample.c        All L13     (C/l Abbrev)----------
Wrote /Users/tnakamur/C_Working/sample.c
```

```
MBP/Users/tnakamur/C_Working 72> cc -o run sample.c -lm
MBP/Users/tnakamur/C_Working 73> ./run
I is 1 J is 2 K is 3
MBP/Users/tnakamur/C_Working 74>
```

実行結果

単純な計算プログラム

- emacs でsample.cを編集　　%emacs sample.c & ↓

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{
  int i,j,k;

  i=1;
  j=2;

  k=i+j;

  printf("I is %d J is %d K is %d\n",i,j,k);


  return 0;

}
```

```
-u:--  sample.c        All L13     (C/l Abbrev)----------
Wrote /Users/tnakamur/C_Working/sample.c
```

```
MBP/Users/tnakamur/C_Working 72> cc -o run sample.c -lm
MBP/Users/tnakamur/C_Working 73> ./run
I is 1 J is 2 K is 3
MBP/Users/tnakamur/C_Working 74>
```

実行結果

割り算にしてみる

k=i/j → k= 1 ／ 2 ？

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{
  int i,j,k;

  i=1;
  j=2;

  k=i/j;

  printf("I is %d J is %d K is %d\n",i,j,k);


  return 0;

}
```

-あu:-- sample.c All L12 (C/l Abbrev)----------

(No changes need to be saved)

- 実行結果

```
MBP/Users/tnakamur/C_Working 74> cc -o run sample.c -lm
MBP/Users/tnakamur/C_Working 75> ./run
I is 1 J is 2 K is 0
MBP/Users/tnakamur/C_Working 76>
```

k=i/j :

1．右辺の計算 i/j = １／２=0.5

2．kに計算結果を代入

kは整数型なので、0.5は代入できない。

切り下げた0が代入される！

- 実数の計算にはintでは無く、浮動小数点型doubleの変数を用いる。

実数の表現

$$2.e+0 \Rightarrow 2.0 \times 10^{0} = 2.0$$

$$2.e+3 \Rightarrow 2.0 \times 10^{3}$$

$$1.e-3 \Rightarrow 1.0 \times 10^{-3}$$

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{
  double i,j,k;

  i=1.e+0;
  j=2.e+0;

  k=i/j;

  printf("I is %e J is %e K is %e\n",i,j,k);


  return 0;

}
```

doubleに変更

実数の表現

%e:double型の出力

```
-u:--  sample.c        All L11     (C/l Abbrev)----------
Wrote /Users/tnakamur/C_Working/sample.c
```

- 実行結果

```
MBP/Users/tnakamur/C_Working 8> gcc -o run sample.c -lm
MBP/Users/tnakamur/C_Working 9> ./run
I is 1.000000e+00 J is 2.000000e+00 K is 5.000000e-01
MBP/Users/tnakamur/C_Working 10>
```

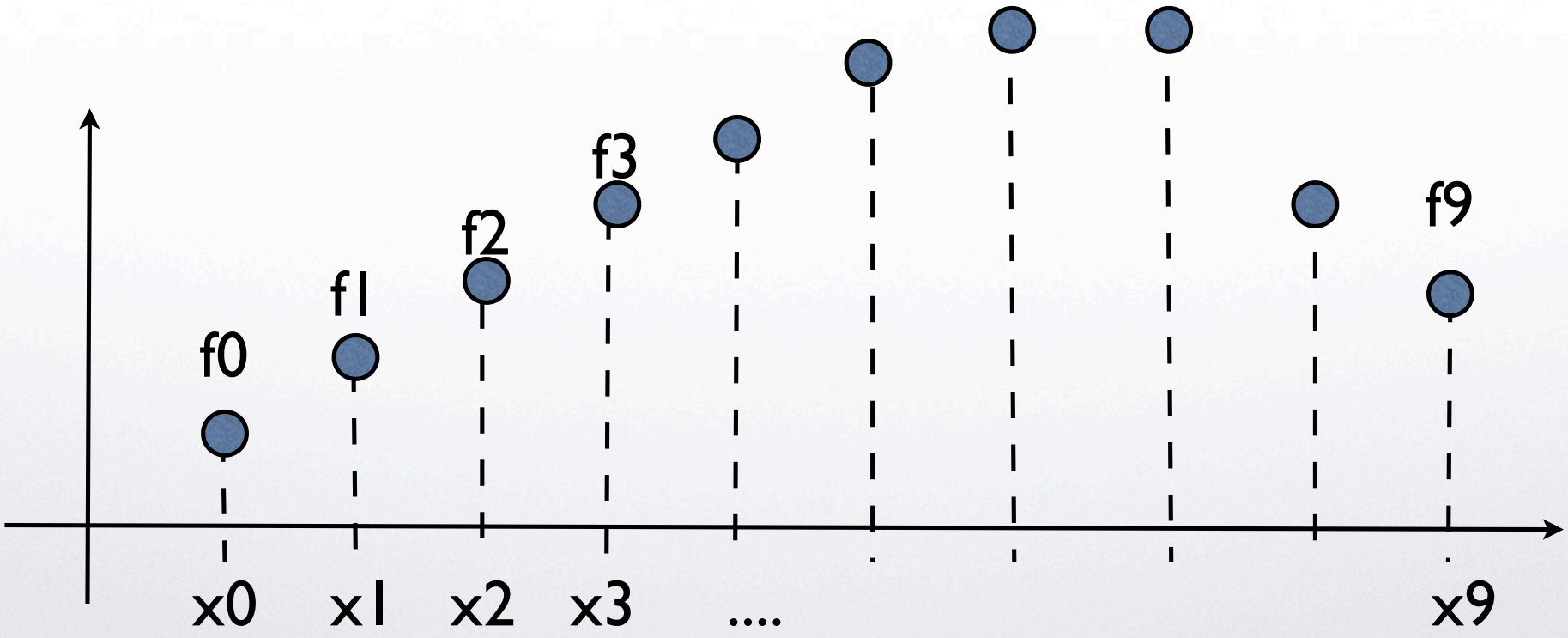
5.0e-1→0.5

配列を使う

- 複数の値を格納する変数：配列

一次元の座標：x0,x1,x2,x3,....,x9

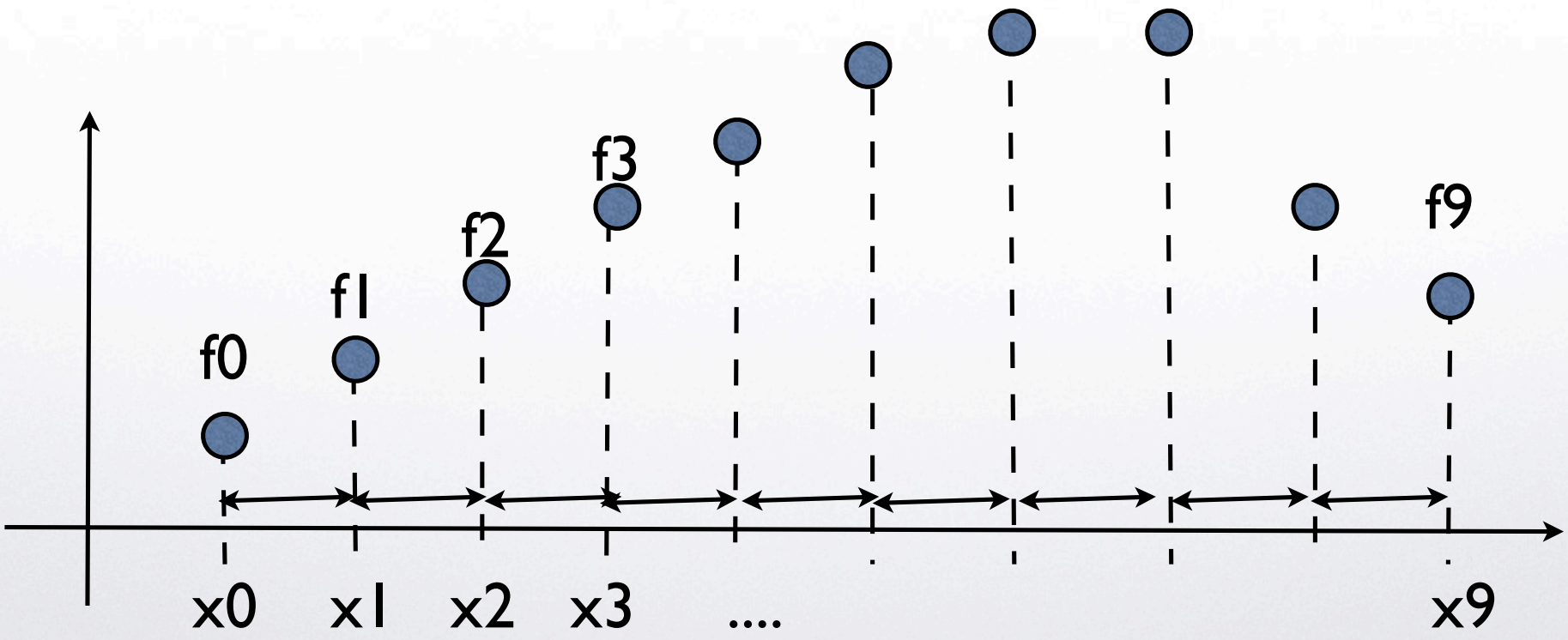
関数の値 ：　　f0, f1, f2, f3 ,....,f9



配列を使う

- 等間隔格子 Δx=2.

$$x_0 = 0, x_1 = \Delta x, x_2 = \Delta x \times 2, x_3 = \Delta x \times 3, \ldots \Rightarrow x_i = \Delta x \times i$$



配列の宣言

double x[10];

double型の変数として、

x[0],x[1],x[2],...,x[8],x[9]

の10個を使えるようにする。

※x[1]〜x[10]では無い！！

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{
  double dx;
  double x[10];

  dx=2.e+0;

  x[0]=dx*0;
  x[1]=dx*1;
  x[2]=dx*2;
  x[3]=dx*3;
  x[4]=dx*4;
  x[5]=dx*5;
  x[6]=dx*6;
  x[7]=dx*7;
  x[8]=dx*8;
  x[9]=dx*9;

  printf("%e\n",x[0]);
  printf("%e\n",x[1]);
  printf("%e\n",x[2]);
  printf("%e\n",x[3]);
  printf("%e\n",x[4]);
  printf("%e\n",x[5]);
  printf("%e\n",x[6]);
  printf("%e\n",x[7]);
  printf("%e\n",x[8]);
  printf("%e\n",x[9]);

  return 0;

}
```

配列の宣言

--:-- sample.c All L32 (C/l Abbrev)----------

Wrote /Users/tnakamur/C_Working/sample.c

実行結果

```
MBP/Users/tnakamur/C_Working 4> cc -o run sample.c -lm
MBP/Users/tnakamur/C_Working 5> ./run
0.000000e+00   x[0]
2.000000e+00   x[1]
4.000000e+00   x[2]
6.000000e+00   x[3]
8.000000e+00   x[4]
1.000000e+01   x[5]
1.200000e+01   x[6]
1.400000e+01   x[7]
1.600000e+01   x[8]
1.800000e+01   x[9]
MBP/Users/tnakamur/C_Working 6>
```

forループを用いた繰り返し

Emacs@MBP-Nakamura.local

```c
#include <stdio.h>
#include <math.h>

int main()
{
  double dx;
  double x[10];

  dx=2.e+0;

  x[0]=dx*0;
  x[1]=dx*1;
  x[2]=dx*2;
  x[3]=dx*3;
  x[4]=dx*4;
  x[5]=dx*5;
  x[6]=dx*6;
  x[7]=dx*7;
  x[8]=dx*8;
  x[9]=dx*9;

  printf("%e\n",x[0]);
  printf("%e\n",x[1]);
  printf("%e\n",x[2]);
  printf("%e\n",x[3]);
  printf("%e\n",x[4]);
  printf("%e\n",x[5]);
  printf("%e\n",x[6]);
  printf("%e\n",x[7]);
  printf("%e\n",x[8]);
  printf("%e\n",x[9]);

  return 0;

}
```

iを1づつ増やして同じ$x_i = \Delta x \times i$の計算

forループを用いた繰り返し

```
for(i=0; i<=9; i++){
    x[i]=dx*i;
}
```

iが9以下である間で、
i=0からiを一つづつ増やしながら
括弧の中を計算する。

Emacs@MBP-Nakamura.local

```
#include <stdio.h>
#include <math.h>

int main()
{
  int i;
  double dx;
  double x[10];

  dx=2.e+0;

  for(i=0; i<=9; i++){
    x[i]=dx*i;
  }

  printf("%e\n",x[0]);
  printf("%e\n",x[1]);
  printf("%e\n",x[2]);
  printf("%e\n",x[3]);
  printf("%e\n",x[4]);
  printf("%e\n",x[5]);
  printf("%e\n",x[6]);
  printf("%e\n",x[7]);
  printf("%e\n",x[8]);
  printf("%e\n",x[9]);

  return 0;

}
```

--:-- sample.c All L16 (C/l Abbrev)----

Emacs@MBP-Nakamura.local

```c
#include <stdio.h>
#include <math.h>

int main()
{
  int i;
  double dx;
  double x[10];

  dx=2.e+0;

  for(i=0; i<=9; i++){
    x[i]=dx*i;
  }

  printf("%e\n",x[0]);
  printf("%e\n",x[1]);
  printf("%e\n",x[2]);
  printf("%e\n",x[3]);
  printf("%e\n",x[4]);
  printf("%e\n",x[5]);
  printf("%e\n",x[6]);
  printf("%e\n",x[7]);
  printf("%e\n",x[8]);
  printf("%e\n",x[9]);

  return 0;

}
```

--:-- sample.c All L16 (C/l Abbrev)----
Wrote /Users/tnakamur/C_Working/sample.c

Emacs@MBP-Nakamura.local

```c
#include <stdio.h>
#include <math.h>

int main()
{
  int i;
  double dx;
  double x[10];

  dx=2.e+0;

  for(i=0; i<=9; i++){
    x[i]=dx*i;
  }

  for(i=0; i<=9; i++){
    printf("%e\n",x[i]);
  }

  return 0;

}
```

--:-- sample.c All L16 (C/l Abbrev)------
Beginning of buffer

宿題

- 等間隔格子: $x_i = \Delta x \times i \quad \Delta x = 2 \quad i = 0,1,2,3,\ldots,100$
- 関数 $f(x) = x^2 + 1 \Rightarrow f[i] = x[i] * x[i] + 1$
- 各格子点の座標と関数の値を以下のように出力するプログラムをx及びfの配列を用いて書きなさい。

0	x[0]	f[0]
1	x[1]	f[1]
2	x[2]	f[2]
3	x[3]	f[3]
	…	

課題：数値積分

$$f(x) = x^2 + 1$$

$$I = \int_0^{200} f(x)dx$$

を矩形／台形／シンプソン則で計算しなさい。

結果は理論値とそれぞれ比較すること。