生態模擬: 以C語言為例

Class 02 (2018/03/08)

- Basic commands in Linux
- Sharing files between ubuntu and host OS (appendix)

- First Step

  1.1 Program in C language
  1.2 Writing codes
  1.3 Programming processes
  1.4 Executing a program

- Basics in C language

  2.1 Basics in C codes
  2.2 Output to computer screen
  2.3 Character, string, and number

Takeshi Miki

三木　健（海洋研究所）

Basic commands in linux

However, we need to learn basic commands to **manage files** with Terminal.

The following is the basic commands in linux.

To check the present directory:	**pwd**
To check the files in the present directly:	**ls** [-a] [-l]
To change the directory place:	**cd** (*relative path or absolute path*)
To read the content of file:	**less** *file_name*
To make a new directory:	**mkdir** *directory_name*
To open or make a text file:	**emacs *file_name***
To copy a file or directory:	**cp** [-r] *file_name_old file_name_new*
To change the name of file:	**mv** *file_name_old file_name_new*
To remove a file or directory:	**rm** [-r] *file_name*

Basic commands in linux: check directory position

To check the present directory:

```
miki$miki-VB-ubuntu:~$  pwd
```

Click *return*, and the directory position is shown.

```
miki$miki-VB-ubuntu:~$  pwd
/home/miki
```

This is the absolute path of the present directory.

Basic commands in linux: list of files and directories

To check the files in the present directory (list):

```
miki@miki-VB-ubuntu:~$ ls
Desktop    Downloads          Music     Public     test.dat
Documents  examples.desktop   Pictures  Templates  Videos
```

Uncolored (white) one represents a file whereas colored (blue) one represents a directory.

There are options in this command to show all files: ls -a

```
miki@miki-VB-ubuntu:~$ ls -a
.                .emacs.d          .gvfs                  .sudo_as_admin_successful
..               .esd_auth         .ICEauthority          Templates
.bash_history    examples.desktop  .local                 test.dat
.bash_logout     .fontconfig       .mozilla               .thumbnails
.bashrc          .gconf            Music                  .vboxclient-clipboard.pid
.cache           .gconfd           .nautilus              .vboxclient-display.pid
.config          .gksu.lock        Pictures               .vboxclient-seamless.pid
.dbus            .gnome2           .profile               Videos
Desktop          .gnome2_private   Public                 .xsession-errors
.dmrc            .gnuplot_history  .pulse                 .xsession-errors.old
Documents        .gstreamer-0.10   .pulse-cookie
Downloads        .gtk-bookmarks    .recently-used.xbel
```

Files with name “.***” are hidden files in which important settings are documented.

Basic commands in linux: list of files and directories

With an option -l, you can check information (e.g. size, propriority, access regulation).

```
miki@miki-VB-ubuntu:~$ ls -l
total 40
drwxr-xr-x 3 miki miki 4096 2011-01-26 13:09 Desktop
drwxr-xr-x 2 miki miki 4096 2011-01-26 11:18 Documents
drwxr-xr-x 2 miki miki 4096 2011-01-26 11:18 Downloads
-rw-r--r-- 1 miki miki  179 2011-01-26 11:00 examples.desktop
drwxr-xr-x 2 miki miki 4096 2011-01-26 11:18 Music
drwxr-xr-x 2 miki miki 4096 2011-01-26 11:18 Pictures
drwxr-xr-x 2 miki miki 4096 2011-01-26 11:18 Public
drwxr-xr-x 2 miki miki 4096 2011-01-26 11:18 Templates
-rw-r--r-- 1 miki miki    8 2011-01-26 11:31 test.dat
drwxr-xr-x 2 miki miki 4096 2011-01-26 11:18 Videos
```

Basic commands in linux: change directory position

When you know the list of files and directories,

```
miki@miki-VB-ubuntu:~$ ls
Desktop    Downloads          Music     Public     test.dat
Documents  examples.desktop   Pictures  Templates  Videos
```

You can change the position by command **cd** *directory_name*:

```
miki$miki-VB-ubuntu:~$  cd Desk
```

Even if you do not completely type the directory name (e.g. Desktop), the file name is complimented with '**tab key**'.

```
miki$miki-VB-ubuntu:~$  cd Desktop/
```

```
miki$miki-VB-ubuntu:~/Desktop$  pwd
/home/miki/Desktop
```

Basic commands in linux: change directory position

miki$miki-VB-ubuntu:~/Desktop$ pwd
/home/miki/Desktop

If you would like to go back *the upper directory*:

```
miki@miki-VB-ubuntu:~/Desktop$ cd ..
miki@miki-VB-ubuntu:~$ pwd
/home/miki
miki@miki-VB-ubuntu:~$
```

You can also use **cd** with *the absolute path*:

miki$miki-VB-ubuntu:~$ cd /home/miki/Desktop

Just with **cd**, you can go back to your **home position** (/home/*user_name*)

```
miki@miki-VB-ubuntu:~/Desktop$ cd
miki@miki-VB-ubuntu:~$ pwd
/home/miki
```

Basic commands in linux: read a file

```
miki@miki-VB-ubuntu:~$ ls -a
.                .emacs.d          .gvfs                .sudo_as_admin_successful
..               .esd_auth         .ICEauthority        Templates
.bash_history    examples.desktop  .local               test.dat
.bash_logout     .fontconfig       .mozilla             .thumbnails
.bashrc          .gconf            Music                .vboxclient-clipboard.pid
```

If you would like to read the file ‘.bashrc’ (*read only*), use **less**:

```
miki$miki-VB-ubuntu:~$  less .bashrc
```

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
[ -z "$PS1" ] && return

# don't put duplicate lines in the history. See bash(1) for more options
# ... or force ignoredups and ignorespace
HISTCONTROL=ignoredups:ignorespace

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000

# check the window size after each command and, if necessary,
# update the values of LINES and COLUMNS.
shopt -s checkwinsize

# make less more friendly for non-text input files, see lesspipe(1)
.bashrc
```

If you would like to go further, you can use cursor.

If you’d like to finish, Just type ‘q’.

Basic commands in linux: make directory and files

You can make a new directory by command **mkdir** *directory_name*:

```
miki$miki-VB-ubuntu:~$  mkdir test
```

Then, make a new file by an editor (e.g. Rstudio)

Basic commands in linux: copy, rename and remove files

Now you have a file ‘test.txt’.

```
miki@miki-VB-ubuntu:~/test.d$ ls
test.txt
```

When you’d like to copy this file to a new file, use **cp** *old_name new_name*:

```
miki@miki-VB-ubuntu:~/test.d$ cp test.txt test2.txt
miki@miki-VB-ubuntu:~/test.d$ ls
test2.txt  test.txt
```

You can change the name of a file by **mv** *old_name new_name*:

```
miki@miki-VB-ubuntu:~/test.d$ mv test2.txt test3.txt
miki@miki-VB-ubuntu:~/test.d$ ls
test3.txt  test.txt
```

You can remove a file by **rm** *file_name*:

```
miki@miki-VB-ubuntu:~/test.d$ rm test3.txt
miki@miki-VB-ubuntu:~/test.d$ ls
test.txt
```

You can remove a directory **rm -r** *directory_name*:

```
miki@miki-VB-ubuntu:~$ rm -r test.d
miki@miki-VB-ubuntu:~$ ls
Desktop    Downloads         Music     Public     test.dat
Documents  examples.desktop  Pictures  Templates  Videos
```

Sharing files between ubuntu and host OS (appendix)

The third step is to '**mount**' this folder to ubuntu with the following command.

```
miki$miki-VB-ubuntu:~$  sudo mount.vboxsf class0_setup /home/miki/shared
–o uid=1000, gid=1000
```

Option of the properiority

Name of Folder in the host OS
You don't need to put absolute path

The absolute path of shared directory in Ubuntu

Space here

What is a program?

For example, MS word asks a computer to display characters, arrange the style, and print it out.

A program asks a computer to do something.

One can make a list of works for a computer.

This is a program, which ask a computer to so a list of works.

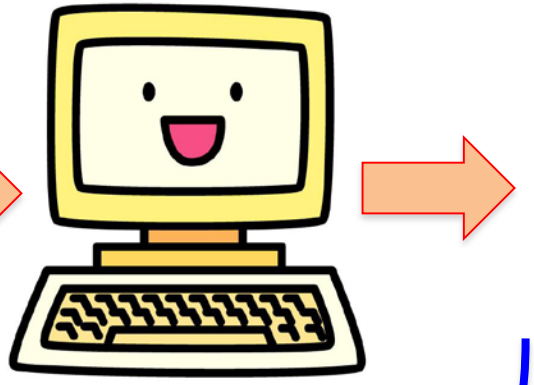
Display,
Print out,
Calculations…

1.1 Program in C language

What is C language?

One can make a list of works for a computer.

This is a program, which ask a computer to so a list of works.



Execution of program

This should be readable by human (= source code).
One language for source code is C language.

Any program is wr
in 'machine code
saved as 'binary f

1.2 Writing codes

A source code is written based on the grammar of C language.

For writing a sample code, use Rstudio and make a file ‘welcome.c’.

```
/*This is a simplest program code
to display “Welcome to …”*/
#include <stdio.h>

int main (void)
{

    //outputting  the string literal to computer screen
    printf(“Welcome to C World! o(^-^)o\n”);

    return 0;
}
```

1.2 Writing codes

A source code is written based on the grammar of C language.

For writing a sample code, use Rstudio and make a file ‘welcome.c’.

```
/*This is a simplest program code
to display “Welcome to …”*/
#include <stdio.h>

int main (void)
{

    //outputting the string literal to computer screen
    printf(“Welcome to C World o(^-^)o\n”);

    return 0;
}
```

Uppercase and lowercase letters are differently recognized (main should not be MAIN)

To start a new line, you need to type ‘return’ key.

Frequency mistake is, “;” and “:’

Frequency mistakes are, “0”(zero) and “O”(alphabet), “1” (number) and “I” (alphabet)

For making empty space, you can use ‘space’ key or ‘tab’ key.

1.3 Programming processes

How to ‘compile’ a source code into a machine code

compile

link

Source file,
written by C-language

Object file,
written by machine codes

Common files
From **library**

Executable file (program),
written by machine codes

compiler
(program)

linker
(program)

gcc
(program: GNU C Compiler)

グニュー

*****.c** → *****.o** → *****.out**

1.3 Programming processes

(1) Compiling process

compile

link

Source file,
written by C-language

Object file,
written by machine codes

Common files
From **library**

Executable file (program),
written by machine codes

*****.c** → *****.o**

```
miki$miki-VB-ubuntu:~$  gcc -c welcome.c
miki$miki-VB-ubuntu:~$  ls
...          welcome.c    welcome.o ← Object file
```

You can read the object file (binary file) with **od**.

```
miki$miki-VB-ubuntu:~$  od welcome.o
```

(2) Linking process

compile

link

Source file,
written by C-language

Object file,
written by machine codes

Common files
From **library**

Executable file (program),
written by machine codes

*****.o** → *****.out**

```
miki$miki-VB-ubuntu:~$  gcc  welcome.o
miki$miki-VB-ubuntu:~$  ls
a.out      welcome.c    welcome.o
```

Executable file

You can indicate any file name for the executable file with option **-o**.

```
miki$miki-VB-ubuntu:~$  gcc welcome.o -o welcome.out
```

1.3 Programming processes

(3) Compiling + linking processes

***.c → ***.out

For simple programs, you do not separate compiling and linking processes.

```
miki$miki-VB-ubuntu:~$  gcc welcome.c -o welcome.out
```

If there are any grammatical mistakes in the source code, terminal will return an error message.

```
miki@miki-VB-ubuntu:~/class_C/class2$ gcc welcome.c -o welcome.out
welcome.c: In function ‘main’:
welcome.c:11: error: expected ‘;’ before ‘return’
miki@miki-VB-ubuntu:~/class_C/class2$ ls
class2_clang.ppt  pics  welcome.c  welcome.c~
miki@miki-VB-ubuntu:~/class_C/class2$
```

In this case, no executable file is made.

1.4 Executing program

In order to execute program, you need to type the name of the program file in Terminal.

```
miki$miki-VB-ubuntu:~$  ./welcome.out
Welcome to C World! o(^-^)o
```

2.1 Basics in C codes

Add one line code to ‘welcome.c’.

```
/*This is a simplest program code
to display “Welcome to …”*/
#include <stdio.h>

int main (void)
{

    //outputting the string literal to computer screen
    printf(“Welcome to C World o(^-^)o\n”);
    printf(“Let’s enjoy learning C language.\n”);
    return 0;
}
```

Then, compile (and link) and execute it.

```
miki$miki-VB-ubuntu:~$  ./welcome.out
Welcome to C World! o(^-^)o
Let’s enjoy learning C language.
```

2.1 Basics in C codes: basic structure of the whole codes

```
/*This is a simplest program code
to display "Welcome to ..."*/
#include <stdio.h>

int main (void)
{

    //outputting the string literal to computer screen
    printf("Welcome to C World o(^-^)o\n");
    printf("Let's enjoy learning C language.\n");
    return 0;
}
```

This is a *comment* for multiple lines.

Include a standard library for display operations

Initial point of the *function* main().

This is a comment for a single lines

First, this is executed.

Second, this is executed.

Finally, this is executed and this is the final point of the function main().

The function main() is called *main function*.

2.1 Basics in C codes: main function and *statement*

Each small process (work for computer) is called ***statement***, which is separated by a **semicolon ';'**.

```
printf("Welcome to C World o(^-^)o\n");
printf("Let's enjoy learning C language.\n");
```

Statement 1

Statement 2

```
.....

int main (void)
{
        A list of statements should be written here.
    return 0;
}
```

Statements in the main function is executed by one statement from the upper to lower lines.

2.1 Basics in C codes: Keep a high readabililty

In C language, spaces between statements are not recognized. So the following source code is also OK (*but you can not write it such as 'm ain ()'*).

```
#include <stdio.h>

int main (void) {   //outputting the string literal to computer screen
printf("Welcome to C World o(^^)o\n"); printf("Let's enjoy learning C language.\n");
      return 0;
}
```

However, it is very important to write codes in a readable way.

```
#include <stdio.h>

int main (void)
{
__//outputting the string literal to computer screen
__printf("Welcome to C World o(^-^)o\n");
__return 0;
}
```

Add comments

Add *indent* by 'tab' key.

2.1 Basics in C codes: Include a necessary library

There are many useful functions (e.g. printf) are prepared in standard libraries in C environment. It is necessary to ***include*** them as ***header files*** before main functions.

stdio.h *for standard input/output*

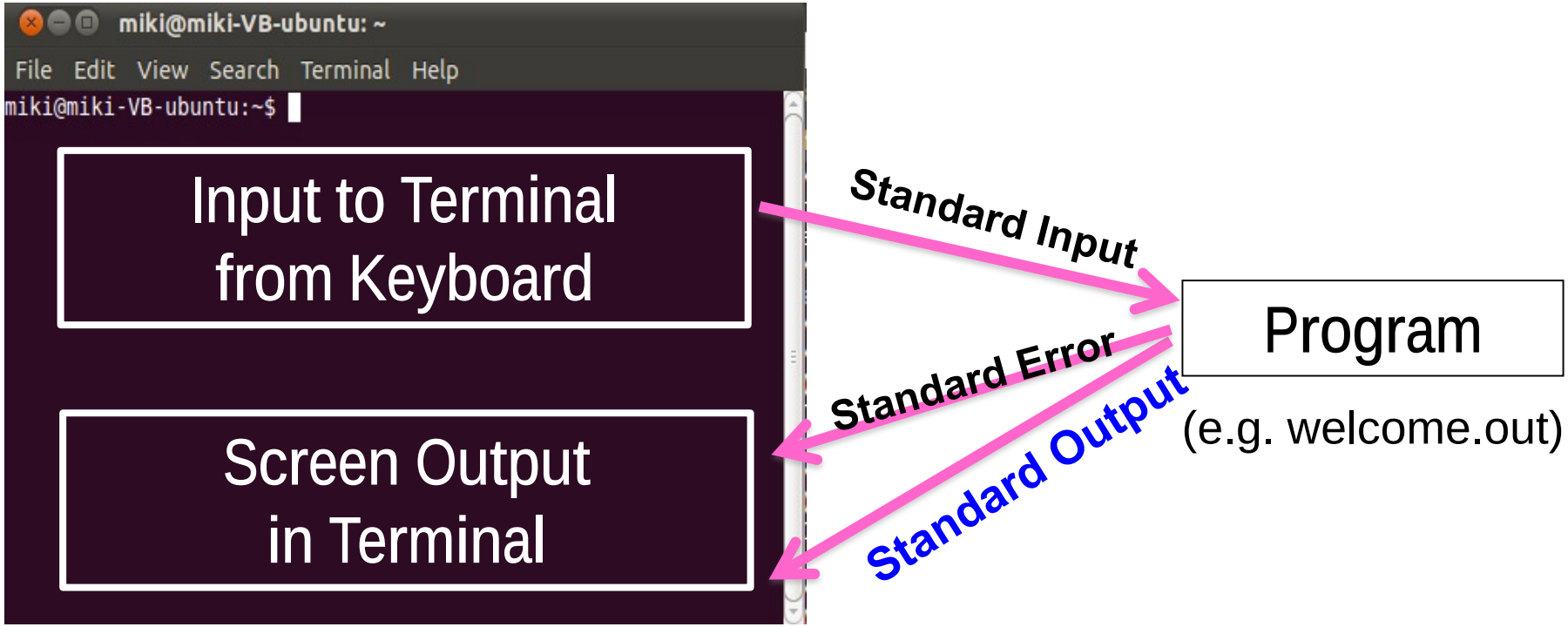
Necessary information for calling (using) printf is stated in this header file.

```
#include <stdio.h>

int main (void)
{
   //outputting the string literal to computer screen
   printf("Welcome to C World o(^-^)o\n");
   return 0;
}
```

2.2 Output to computer screen: Standard output

In the second section, we will learn more about the function **printf** for ***standard output*** **(格 式 化 輸 出 函 數)** of characters and numbers to computer screen.



2.2 Output to computer screen: Standard output

In the second section, we will learn more about the function **printf** for ***standard output*** of characters and numbers to computer screen.

```
....
int main (void)
{

    printf(“                    ”);
        ...
    return 0;
}
```

A string of characters for outputting computer screen, which is called *string literal*.

2.2 Output to computer screen: Start a new line in output

An *escape sequence*, **\n**, is necessary for starting an new line (新行) in the standard output.

```
int main (void)
{

    printf("Welcome to C World o(^-^)o\n");
    printf("Let's enjoy learning C language.\n");
    …
```

Try to check what happens in the standard output if you do not add '\n' in the first statement.

```
….
int main (void)
{

    printf("Welcome to C World o(^-^)o");
    printf("Let's enjoy learning C language.\n");
    …
```

2.2 Output to computer screen: Add a tab

Another frequently used *escape sequence*, **\t**, is necessary for adding 'tab' (跳格) after the sequence of characters.

```
int main (void)
{

    printf("Welcome to C World o(^-^)o\t");
    printf("Let's enjoy learning C language.\n");
    …
```

This will be highly useful for outputting number data for reading it by other programs, e.g. gnuplot, R, or excel.

You can also try this.

```
int main (void)
{

    printf("\\(^o^)/\n");
    …
```

2.3 Character, string, and number: Conversion Specification

Conversion Specification (轉換字元)

When you would like to output numbers as the standard output, it is necessary to specify the *type* of output for a program to recognize it.

Later, we need to distinguish ***character*** (= a single character) [字元], ***numerical value*** (integer and float)[整數,浮點數], and ***string*** (= a string of characters) [字串].

Prepare another source file:

```
#include <stdio.h>

int main (void)
{
    printf("%c is a character. \n", 'A');
    printf("%d is an integer. \n", 123);
    printf("%f is a floating point number. \n", 10.5);
    printf("%.4f is a floating point number.\n", 10.5);

    return 0;
}
```

Output of a character

Output of an integer

Output of a floating point number

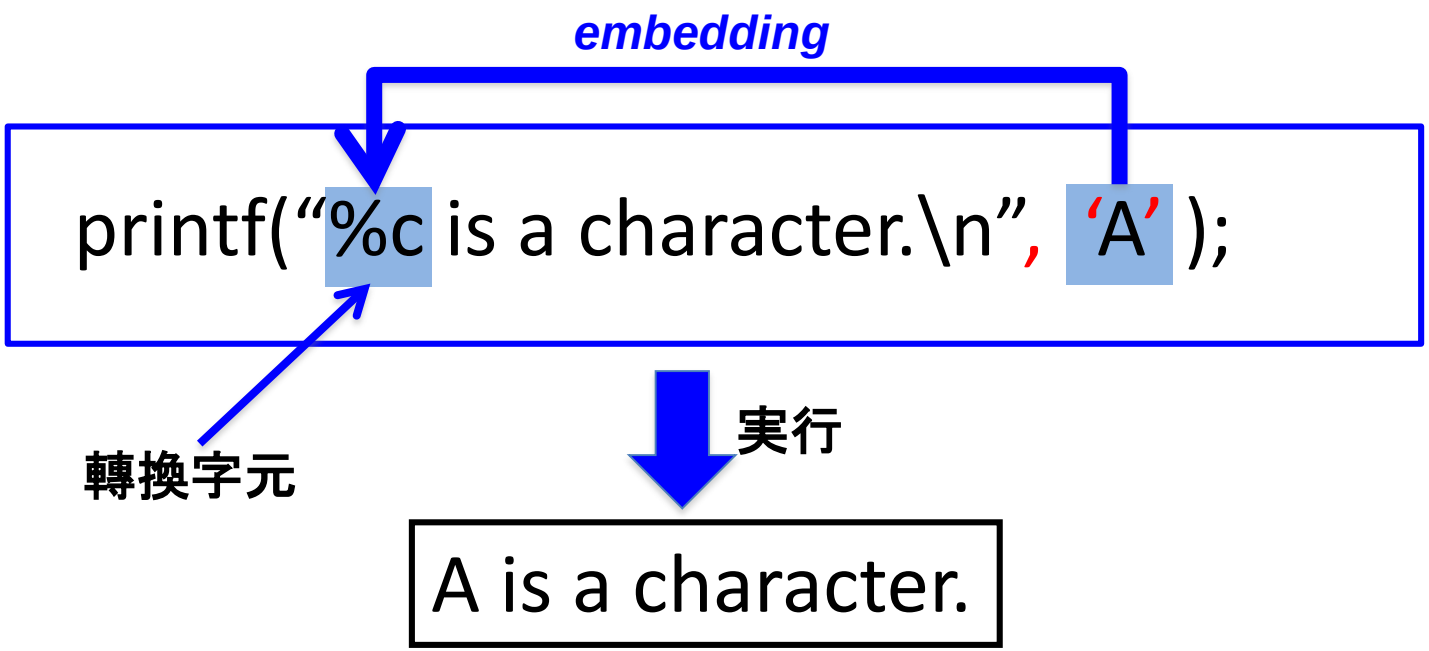
You can specify the number of figures.

2.3 Character, string, and number: Conversion Specification

The output of this program should be:

```
miki@miki-VB-ubuntu:~/class_C/class2$ gcc output_test.c -o output.out
miki@miki-VB-ubuntu:~/class_C/class2$ ./output.out
A is a character.
123 is an integer.
10.500000 is a floating point number.
10.5000 is a floating point number.
```

You can *embed* characters/numbers by using 轉換字元.



2.3 Character, string, and number: Numerical values

In C, integer and floating point number are distinguished.

Integer constant: 1, 3, 100…
Floating constant: 2.1, 3.14, 5.0…

```
printf("%d is a integer.\n", 123 );
```

```
printf("%f is a float.\n", 10.5 );
```

```
printf("%.4f is a float.\n", 10.5 );
```

2.3 Character, string, and number: application

You can also output multiple characters/numerical values.

Prepare another source file:

```
#include <stdio.h>

int main (void)
{
    printf("%c is a character, %d is an integer, and %.2f is a float. \n", 'A', 123, 10.5);

    return 0;
}
```

実行

```
A is a character, 123 is an integer, and 10.50 is a float.
```

Homework this week

(1) Correct the following source codes and arrange them in a readable way.

```
#include <stdio.h>
int main (void) {Printf(“How are you? \n”); printf(“Not bad”\n):
return O;
```

(2) Write a source code, which is intended to output the following tab separated numbers.

```
1.2	3.4	0.5
3.5	6.0	-10
```