COMP2006 Computer Organization
Introduction to C Language

Part I: Write your 1st C program

Now we are going to learn how to write and run your C program on your computer. There are three steps:

1. Creating the source file
2. Compiling the program
3. Executing the program

Creating the source file

To start writing your code, you need a text editor or IDE (integrated development environment). There are many different editors we can use, such as Nodepad++, Visual Studio Code(VS Code), Eclipse for C++, etc.

In this lab, we use VS Code. You can download it through the following link:

https://code.visualstudio.com/download

Let us launch VS Code, press Ctrl + N to create a new file. Then, type the following program code in it.

```c
#include <stdio.h>
int main()
{
   printf("Hello World!\n");
   return 0;
}
```

The C code above prints a string “Hello World” in the console. It equivalents the following Processing statement.

```
println("Hello World!");
```

Then, press CTRL+S to save the file on the Desktop and name the file “helloworld.c”.

Compiling and Executing Program

The code is now human-readable but not a computer. So, we need to compile the code written in C before it can be run on the computer. Compiling is the process of converting the code from human-readable into machine-readable instructions so that the computer can run it.

When you compile your source file, a new compiled file is created. To compile a C program, we need a C compiler.

If you use Microsoft Windows, you may use TDM GCC. If you use Macintosh, you may use HOMEBREW GCC. Please find the document named “Installing GCC for Home PC” if your GCC is not ready.

Launch the terminal (click VS Code’s menu “Terminal” > “New Terminal”) and enter the command below will compile “helloworld.c” into a new file called “firstprogram”:

```
> gcc helloworld.c -o firstprogram
```

If you do not see any error message after compiling the C code, you can now run the executable by entering the name of the executable in the terminal. Otherwise, you need to correct your program code, save the file and compile it again.

Windows:

```
> firstprogram
```

Mac:

```
> ./firstprogram
```

Compiling C Program to Assembly Code

We can compile the C programs to assembly code by using *-S* option. The *-S* option means that compile only but does not assemble and link.

Let us use the helloworld.c as an example to try the *-S* option.

```
> gcc helloworld.c -S
```

The command above generates a new file named “helloworld.s”. You can open the file using any text editor. The following is the content of the file:

```
   .file "helloworld.c"
   .text
   .def  __main;     .scl  2;    .type 32;   .endef
   .section .rdata,"dr"
.LC0:
   .ascii "Hello World!\0"
   .text
   .globl      main
   .def  main; .scl  2;    .type 32;   .endef
   .seh_proc   main
main:
   pushq %rbp
   .seh_pushreg      %rbp
   movq  %rsp, %rbp
   .seh_setframe     %rbp, 0
   subq  $32, %rsp
   .seh_stackalloc   32
   .seh_endprologue
   call  __main
   leaq  .LC0(%rip), %rcx
   call  puts
   movl  $0, %eax
   addq  $32, %rsp
   popq  %rbp
   ret
   .seh_endproc
   .ident      "GCC: (tdm64-1) 9.2.0"
   .def  puts; .scl  2;    .type 32;   .endef
```

Formatting Output with Printf

The **printf** function is one of many functions provided in the C Standard Library *<stdio.h>*. It causes the computer to perform an action to output a string to the console screen (default output device). It works like the **print** and **println** functions in Processing.

The backslash "\" is called an escape character. It indicates that **printf** is supposed to do something out of the ordinary. When encountering a backslash in a string, the compiler looks ahead at the next character and combines it with the backslash to form an escape sequence.

Some Common Escape Sequences:

\n	Newline – move the cursor to the beginning of the next line.
\t	Tab – move the cursor to the next tab stop.
\\	Backslash.
\"	Double-quote.

```
#include <stdio.h>

int main() {
    printf("Hello\tWorld!\n");
    printf("Hello \"World\"\n");
    printf("Hello \\World\\\n");

    return 0;
}
```

Output

```
Hello   World!
Hello "World"
Hello \World\
```

Printing Numbers

Together with the percent sign (*%*), these form conversion specifications. The **printf** function can perform the formatting capabilities, such as rounding floating point values, displaying data with fixed-size field widths, etc. The **printf** function has the form:

```
printf(format-control-string, other-arguments);
```

The commonly used conversion specifications are:

%d	Display a signed decimal integer.
%ld	Display a signed long integer.
%f	Display a signed floating-point number.
%lf	Display a signed long double floating-point number.

Compile and run the following C program

```c
#include <stdio.h>

int main() {
   double a = 5;
   double b = 3;
   double c;

   c = a + b;
   printf("a + b = %f\n", c);
   c = a - b;
   printf("a - b = %f\n", c);

   printf("a * b = %f\n", a*b);
   printf("a / b = %f\n", a/b);
   return 0;
}
```

The Processing version:

```
double a = 5;
double b = 3;
double c;

c = a + b;
println("a + b = ", c);

c = a - b;
println("a - b = ", c);

println("a * b = ", a * b);
println("a / b = ", a / b);
```

This example code above shows how to:

- Use variables which are a, b, and c.
- Assign values to the variables.
- Display formatted strings.

Printing Strings and Characters

The *%s* and *%c* are used to print individual strings and characters respectively. The *%s* conversion specification requires a reference of a char array as an argument. The %c conversion specification requires a char argument. Note that we use a char array to store a string in the C program.

Compile and run the following C program to check its output:

```c
#include <stdio.h>

int main() {
    char a = 'A';
    char b[] = "This is a string";

    printf("%c\n", a);
    printf("%c\n", b[0]);
    printf("%s\n", b);
    printf("%s\n", "This is also a string");

    return 0;
}
```

Strings in C

As mentioned that C uses a char array to represent a string. If we run the following line to create a string "HELLO WORLD", a char array of size *12* will be created in the system memory. And, it stores the characters following by a **NULL** character ('\0'). The **NULL** character represents the end of the string.

```
char str[] = "HELLO WORLD";
```

The variable name **str** stores the reference (the starting memory) of the char array.

0	1	2	3	4	5	6	7	8	9	10	11
H	E	L	L	O		W	O	R	L	D	\0

Let's try the following program and see whether you know what it is done.

```
#include <stdio.h>

int main() {
    char str[] = "HELLO WORLD";
    str[1] = str[1] - 'A' + 'a';
    printf("%s\n", str);
    return 0;
}
```

Printing with Field Widths and Precisions

The exact size of a field in which data is printed is specified by a field width. If the field width is larger than the data being printed, the data will normally be right-justified within that field. The precision specifies the number of digits to appear after the decimal point when a floating-point conversion specification is used.

In the following program, the **printf** functions with the field width of 30 to format the outputs.

```
#include <stdio.h>

int main() {
    printf("%30s\n", "Hi all!");
    printf("%30s\n", "Welcome to COMP2006 lab");
    printf("%30s\n", "C is fun!");

    return 0;
}
```

In the following program, the **printf** functions print the floating-point numbers with different precisions.

```c
#include <stdio.h>

int main() {
    float pound = 2.20462f;
    double pi = 3.14159265359;

    printf("1 kg = %.2f pounds\n", pound);
    printf("PI = %.11lf\n", pi);

    return 0;
}
```

Compile C Program on Linux

How about we compile the C program on Linux platform? Let us connect and log into CSLINUX1 and try it out.

1. Connect CSLINUX1 server:
   - Windows user, use PuTTY with the following information:
     i. Host name: *cslinux1.comp.hkbu.edu.hk*
     ii. Port: *22*
   - macOS user, use the following command:
     ```
     ssh -l e######## cslinux1.comp.hkbu.edu.hk
     ```

2. Download the C program using the following command:
   ```
   wget http://www.comp.hkbu.edu.hk/~mandel/comp2006/helloworld.c
   ```

3. Compile the C program:
   ```
   gcc helloworld.c -o helloworld
   ```

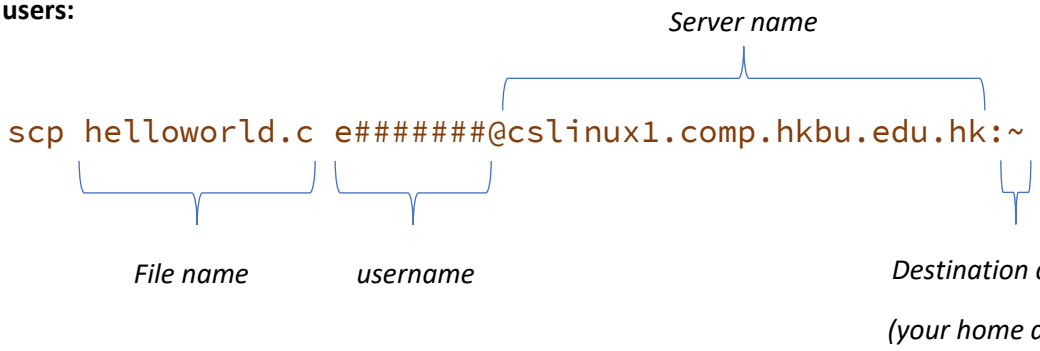
4. Add the execution permission to the output file:
   ```
   chmod a+x helloworld
   ```

5. Run the compiled program:
   ```
   ./helloworld
   ```

Uploading File to CSLINUX1

You may update your own C program to CSLINUX1 and compile it by using SCP command (macOS users) or WinSCP (Windows users).

macOS users:

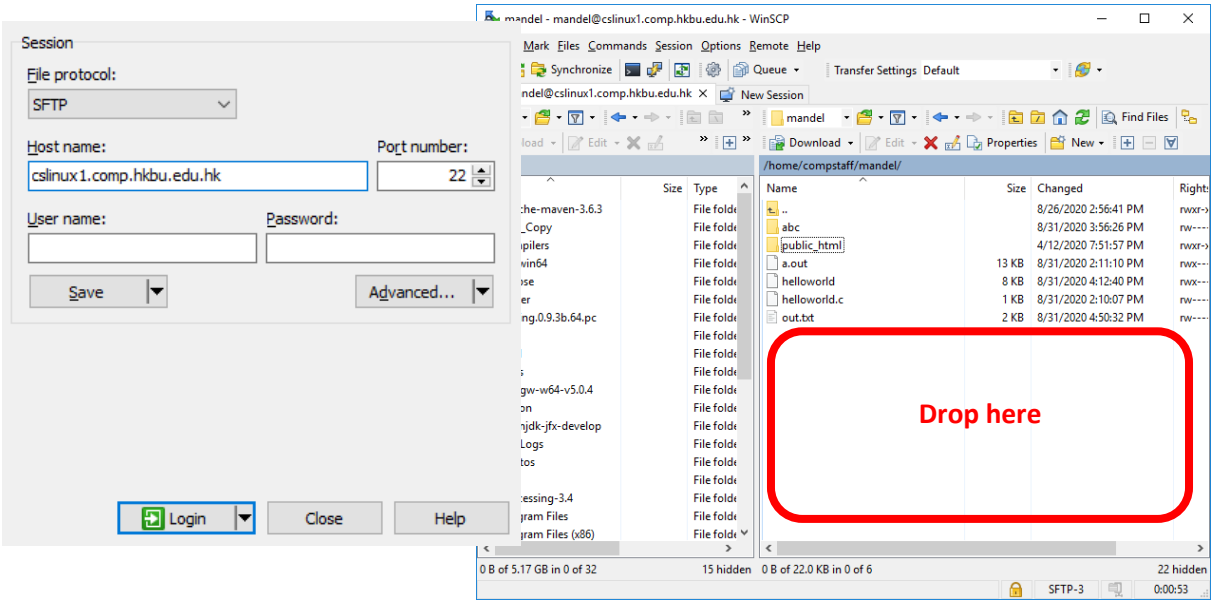


Windows users:

1. Download WinSCP through the following link:
   https://winscp.net/download/WinSCP-5.17.7-Setup.exe

2. Double-click the installation file to install WinSCP.

3. Launch WinSCP and fill *cslinux1.comp.hkbu.edu.hk* and *22* respectively for the host name and port.



4. Click “Login” to connect the server.

5. Input your username and password to log in.

6. To upload, drag the file (files/directories) from Windows Explorer to the right panel.

Exercise

Exercise 1

Given an incomplete program code below:

```c
#include <stdio.h>

int main() {

    float radius = 7.5f;
    double pi = 3.14159265359;
    char circle = 'A';

    printf(_____________, ________, _____);
    printf(_____________, ________, ________);
    printf(_____________, ________, ________);

    return 0;
}
```

Complete the program code so that it outputs as follows:

```
The radius of Circle A is 7.50
The area of Circle A is 176.71
The circumference of Circle A is 47.12
```

Exercise 2

Given an incomplete program code below:

```c
#include <stdio.h>

int main() {
    char f1[] = "name";
    char f2[] = "size";
    char f3[] = "location";
    char v1[] = "appointment.xlsx";
    float v2 = 512332;
    char v3[] = "home/pi/Desktop/";
    ...
    return 0;
}
```

Complete the program code so that it outputs as follows:

```
      name : appointment.xlsx
      size : 500.32KB
  location : home/pi/Desktop/
```