

System Software

Introduction to course

Jarno Tuominen, Spring 2024

Course Practises

- Schedule
- Exercises
- Grading



Schedule

Lectures mainly on Fridays (a few Wednesdays as well)

1 hour theory part (lecture)

2 hours demo/exercise session

- Occasionally teacher may (or may not) show some demonstration
- Mainly you will be working with the exercises, while teacher is supporting you on a need basis

Presence is mandatory - no online/ recorded lectures
Communications over Teams - feel free to ask help!



Exercises

Two systems in use

- Ville for automatically graded exercises
- Virtual machine-based development environment on your own PC
- Git for submitting your work

Note: ItsLearning not in use. All the material in Git. Follow also Teams.



Grading

Based on:

- Completion rate of exercises
- Git activity
- A short exam testing basic concepts

Weighting to be decided (later on)



Questions?



System Software Basics of C Language

Sanna Määttä

C Language

C Language

Originally C was designed for UNIX operating system.

C was created from ALGOL, BCPL and B programming languages and was first published in 1972.

Latest C standard is from 2018, ISO/IEC 9899:2018 (C17 is the informal name).

- The ISO/IEC standard is not free.
- You can find a final draft version of the 2011 standard here: <http://www.open-std.org/jtc1/sc22/wg14/www/docs/n1570.pdf>



C Language

C is a general purpose and procedural programming language.

C has a static type system.

- This is different compared to e.g. Python, where you do not always need to define the type of a variable.



Where Is C Language Used?

Embedded Systems, IoT applications

In System Software (operating system kernels, device drivers, Bootloader, BIOS/UEFI, system applications...)

Most operating systems (at least their kernels) are written in C (UNIX, Linux, Windows, MacOS, iOS, Android, Windows Phone).

Most databases (Oracle DB, MySQL, PostgreSQL, MS SQL Server) are written in C/C++.



First Program in C

How to Start Writing Code

Write the “Hello World” program first.

Then compile and run your code, this way you know your development environment is working properly!

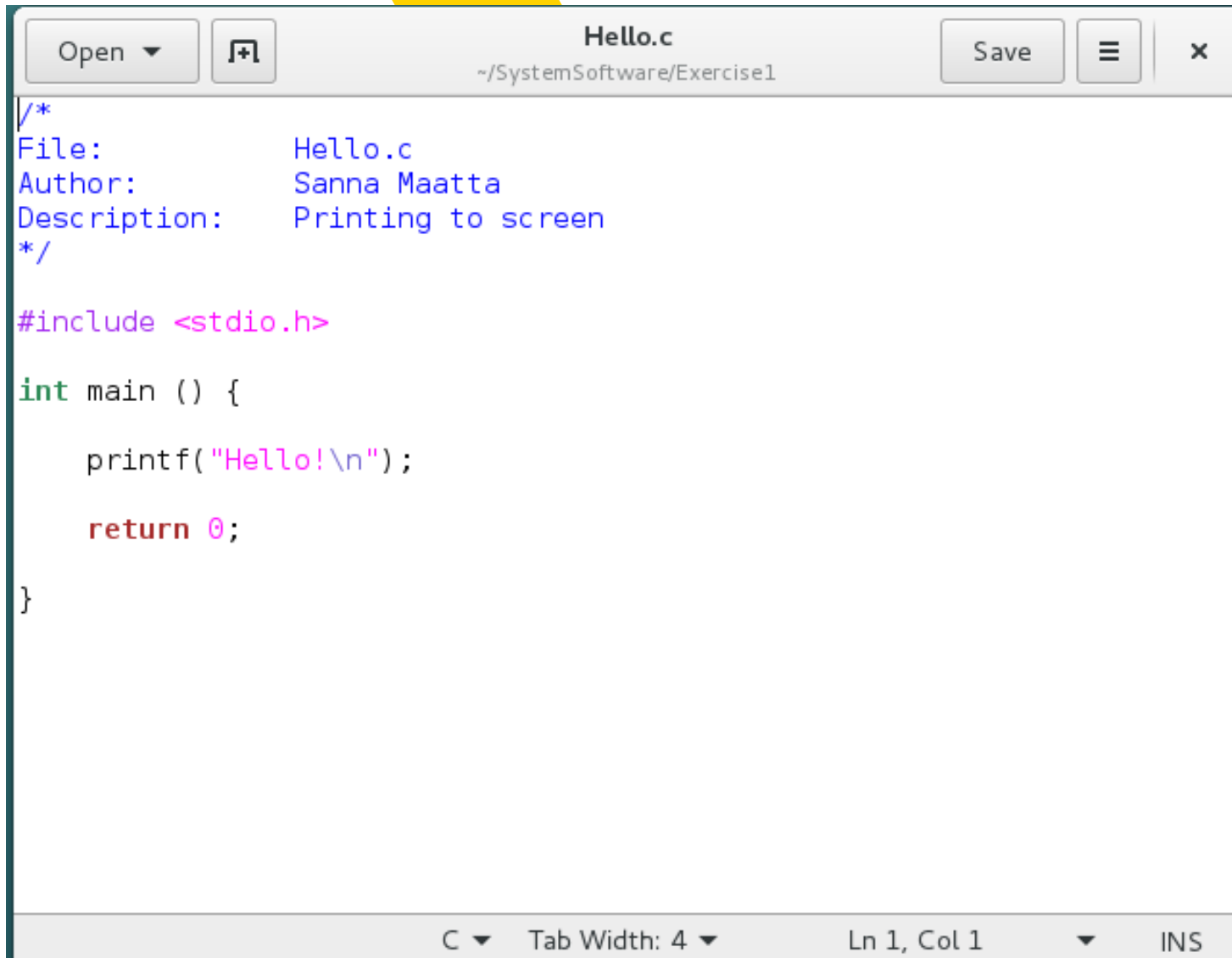


```
File: Hello.c
Author: Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

The screenshot shows a code editor window with the title 'Hello.c' and a path '~/.SystemSoftware/Exercise1'. The code is a C program that prints 'Hello!' to the screen. The code is color-coded: comments are blue, preprocessor directives are magenta, and keywords are green. The status bar at the bottom shows 'C', 'Tab Width: 4', 'Ln 1, Col 1', and 'INS'.



```
/*
File:      Hello.c
Author:    Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

For comparison, same in Python.

```
1 # File: main.py
2 # Author: Sanna Maatta
3 # Printing to screen
4
5 print("Hello")
```

First Program in C

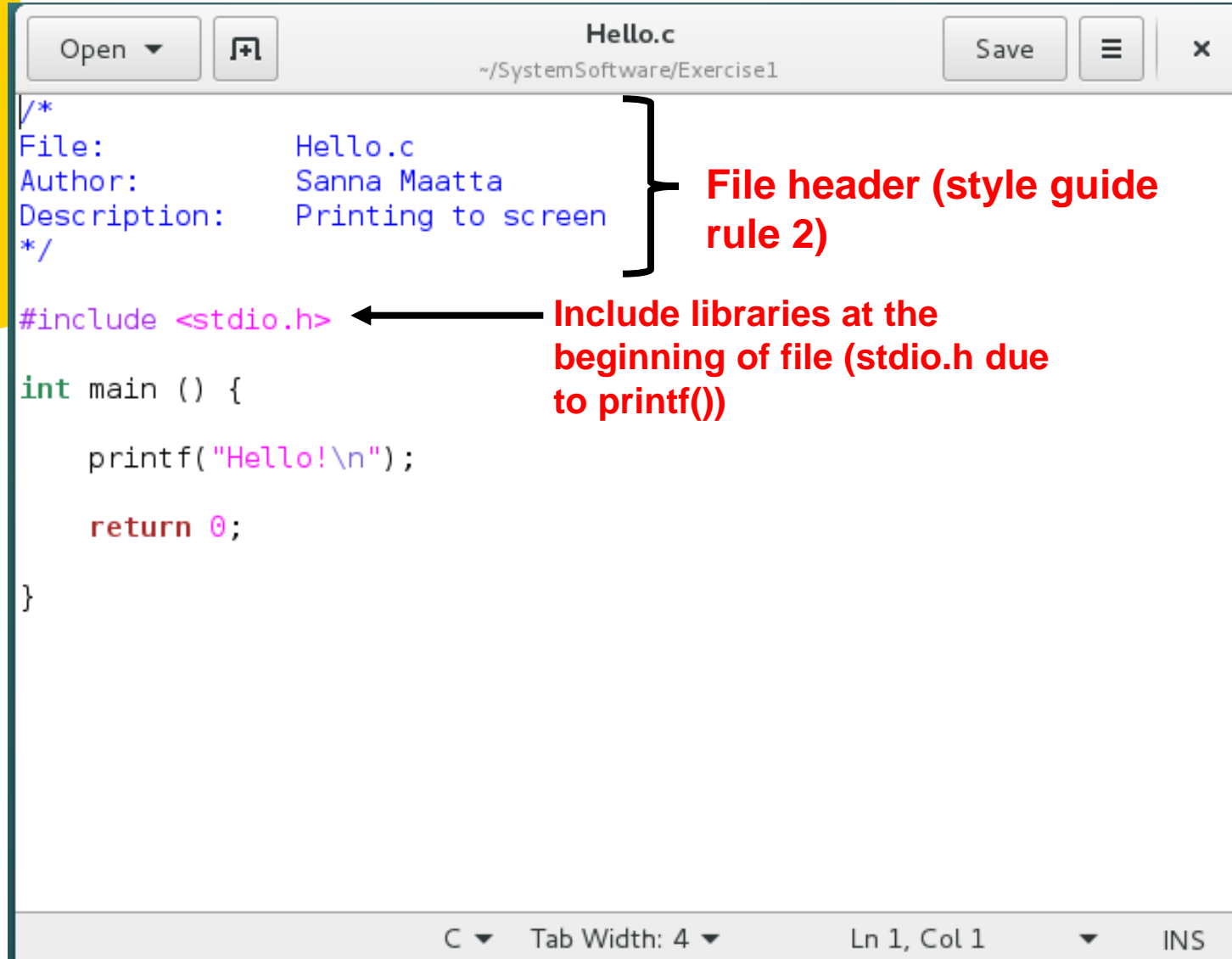
```
Open ▾ [icon] Hello.c ~/SystemSoftware/Exercise1 Save [menu] x
/*
File:      Hello.c
Author:    Sanna Maatta
Description: Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

} File header (style guide rule 2)

C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

First Program in C



The screenshot shows a code editor window titled "Hello.c" with the path "~/SystemSoftware/Exercise1". The code is as follows:

```
/*  
File:          Hello.c  
Author:       Sanna Maatta  
Description:   Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
    return 0;  
}
```

Annotations on the right side of the code:

- A bracket groups the multi-line comment header, with the text: **File header (style guide rule 2)**
- An arrow points to the `#include <stdio.h>` line, with the text: **Include libraries at the beginning of file (stdio.h due to printf())**

The status bar at the bottom indicates: C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

First Program in C

Main function
Each C program shall
have 1 main function.

```
File: Hello.c
Author: Sanna Maatta
Description: Printing to screen
*/
#include <stdio.h>
int main () {
    printf("Hello!\n");
    return 0;
}
```

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf()).

Main function

First Program in C

Main function
Each C program shall
have 1 main function

```
Open ▾ [Icon] Hello.c ~/SystemSoftware/Exercise1 Save [Menu] x

/*
File:      Hello.c
Author:    Sanna Maatta
Description: Printing to screen
*/
#include <stdio.h>
int main () {
    printf("Hello!\n");
    return 0;
}
```

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf())

Prints Hello!

C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

First Program in C

Main function
Each C program shall
have 1 main function

```
File: Hello.c
Author: Sanna Maatta
Description: Printing to screen
*/
#include <stdio.h>
int main () {
    printf("Hello!\n");
    return 0;
}
```

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf())

Prints Hello!

Each *non void* function returns something.

C ▾ Tab Width: 4 ▾ Ln 1, Col 1 ▾ INS

First Program in C

Main function
Each C program shall
have 1 main function

```
File: Hello.c
Author: Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf())

Prints Hello!

Each *non void* function returns something

Indentation (style guide rule 13)

Tab Width: 4

First Program in C

Main function
Each C program shall
have 1 main function

File naming (style guide rule 5)

```
File: Hello.c
Author: Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

File header (style guide rule 2)

Include libraries at the beginning of file (stdio.h due to printf())

Prints Hello!

Each *non void* function returns something

Indentation (style guide rule 13)

Tab Width: 4

First Program in C

```
File: Hello.c
Author: Sanna Maatta
Description: Printing to screen
*/
#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

Functions' statements are enclosed in curly brackets { }

Semicolon ; finishes each sentence

C ▼ Tab Width: 4 ▼ Ln 1, Col 1 ▼ INS

First Program in C

```
File: Hello.c
Author: Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

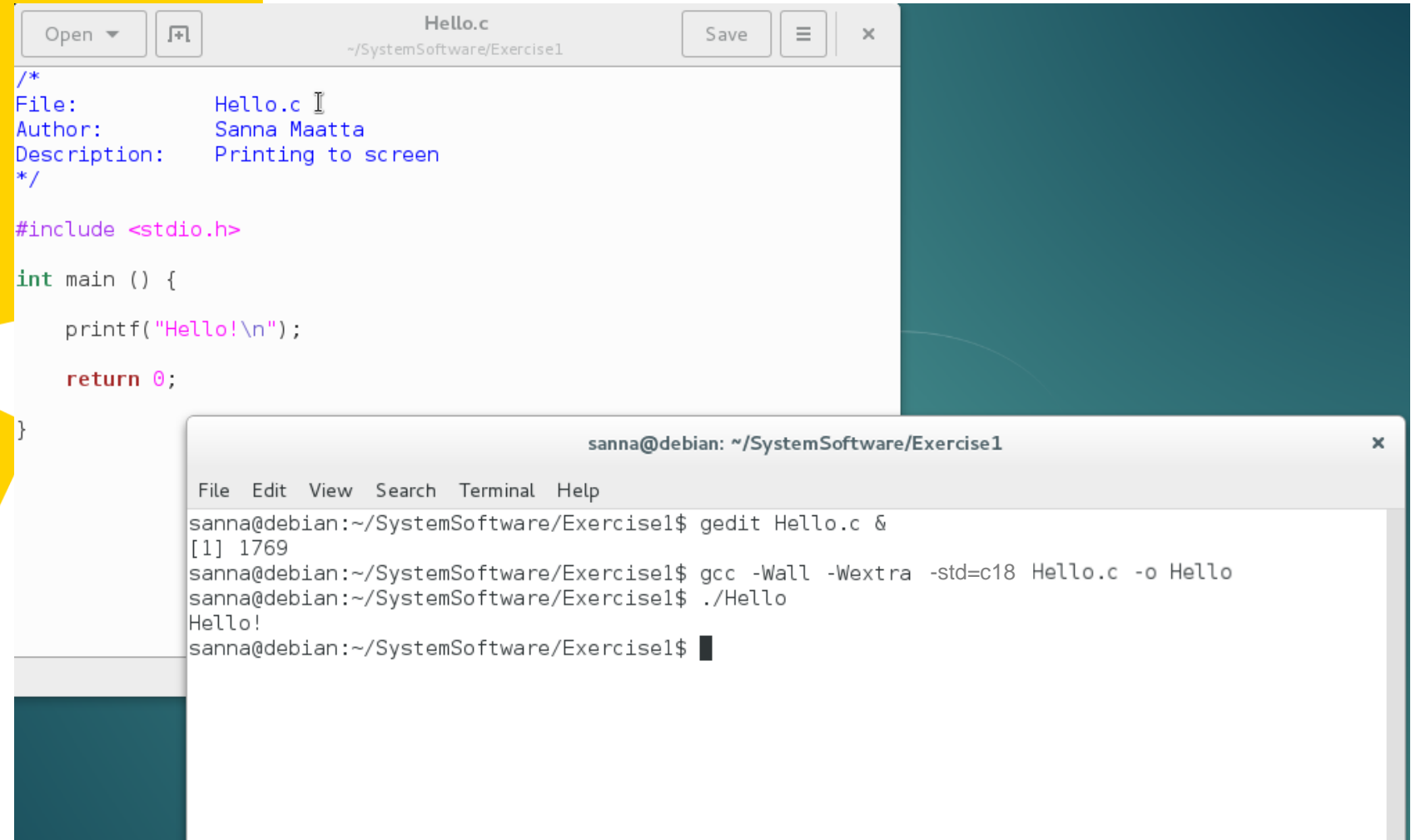
int main () {
    printf("Hello!\n");
    return 0;
}
```

Pre-processor commands start with #

C Tab Width: 4 Ln 1, Col 1 INS

How to Compile and Run C Code on Linux

Compiling and Running on Linux



The screenshot displays a Linux desktop environment. In the background, a code editor window titled 'Hello.c' is open, showing the source code for a C program. The code includes a comment block with file information and a main function that prints 'Hello!' to the screen. In the foreground, a terminal window titled 'sanna@debian: ~/SystemSoftware/Exercise1' is open, showing the commands used to compile and run the program. The terminal output shows the program's execution and the resulting 'Hello!' message.

```
/*
File:      Hello.c
Author:    Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

int main () {

    printf("Hello!\n");

    return 0;
}
```

```
sanna@debian: ~/SystemSoftware/Exercise1
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

Compiling and Running on Linux



The screenshot shows the Gedit editor window titled 'Hello.c' with the following content:

```
/*
File:      Hello.c
Author:    Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

int main () {
    printf("Hello!\n");
    return 0;
}
```

**Open/create file "Hello.c"
in Gedit editor**

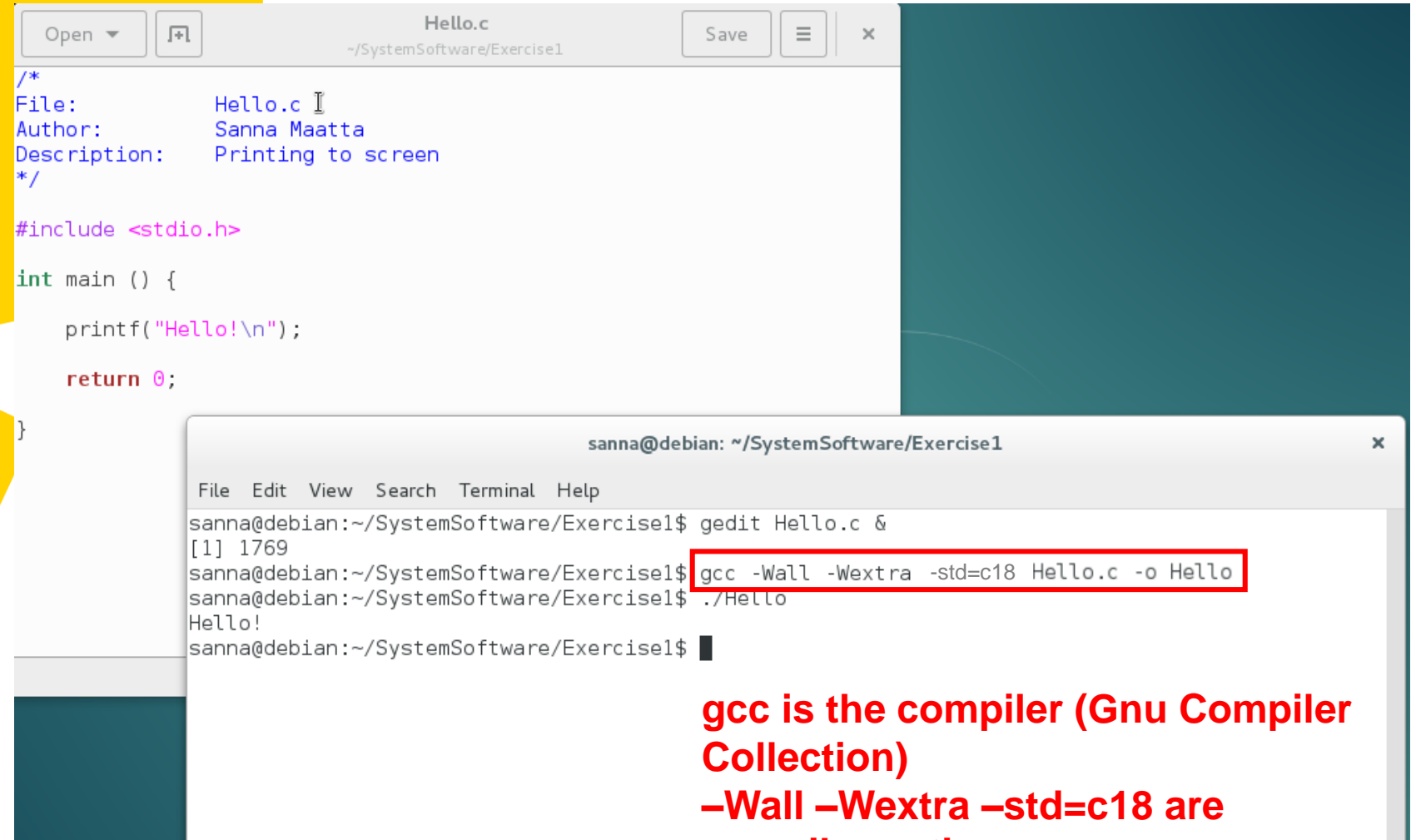


The screenshot shows a terminal window titled 'sanna@debian: ~/SystemSoftware/Exercise1' with the following commands and output:

```
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

**& (ampersand) is not
mandatory but allows you
to use terminal while file
is open, so use it!**

Compiling and Running on Linux



The screenshot displays a Linux desktop environment. In the background, a code editor window titled 'Hello.c' is open, showing the following C code:

```
/*
File:      Hello.c
Author:    Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

int main () {

    printf("Hello!\n");

    return 0;
}
```

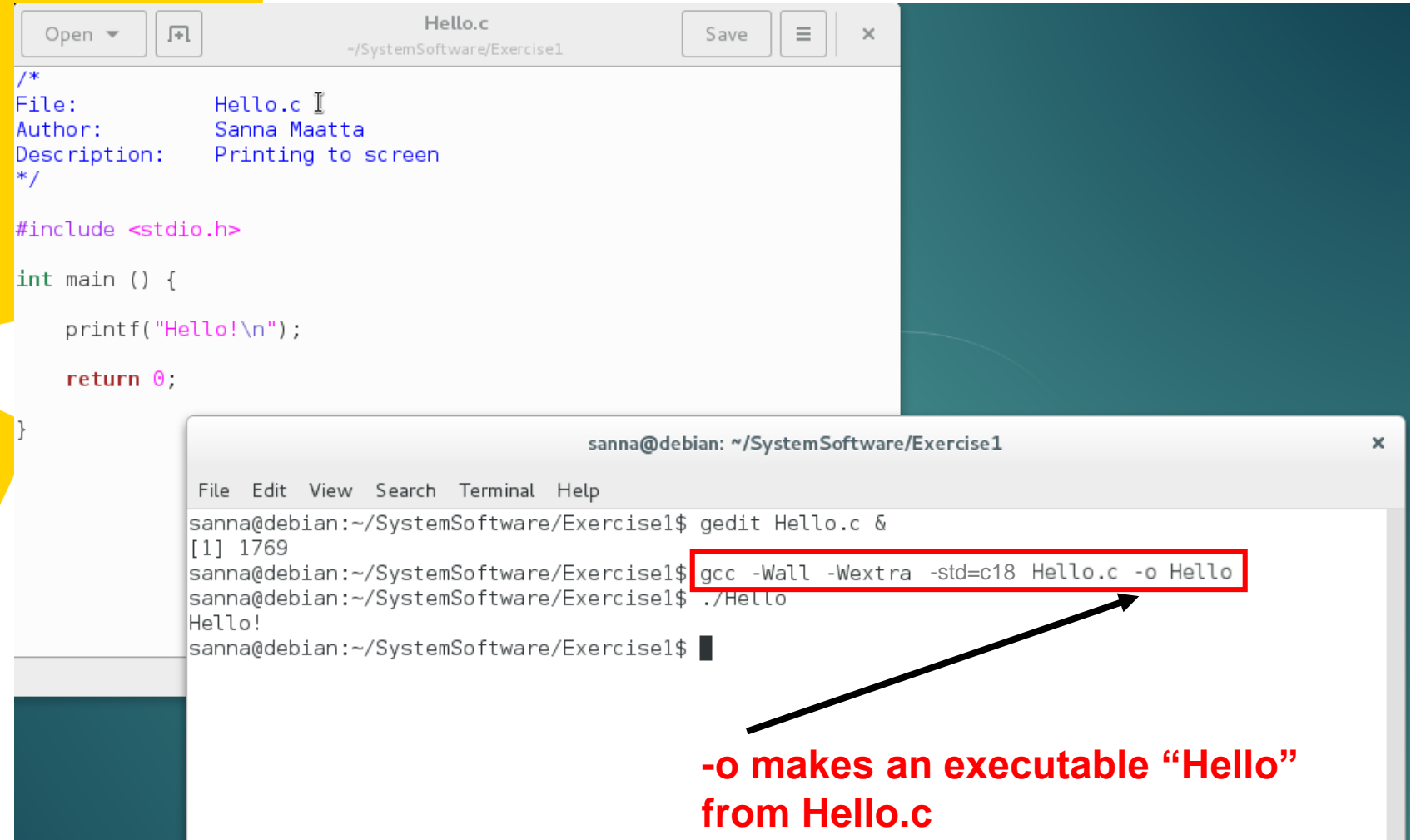
In the foreground, a terminal window titled 'sanna@debian: ~/SystemSoftware/Exercise1' is open. It shows the following commands and output:

```
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

The command `gcc -Wall -Wextra -std=c18 Hello.c -o Hello` is highlighted with a red rectangle in the terminal.

gcc is the compiler (Gnu Compiler Collection)
-Wall -Wextra -std=c18 are compiler options

Compiling and Running on Linux



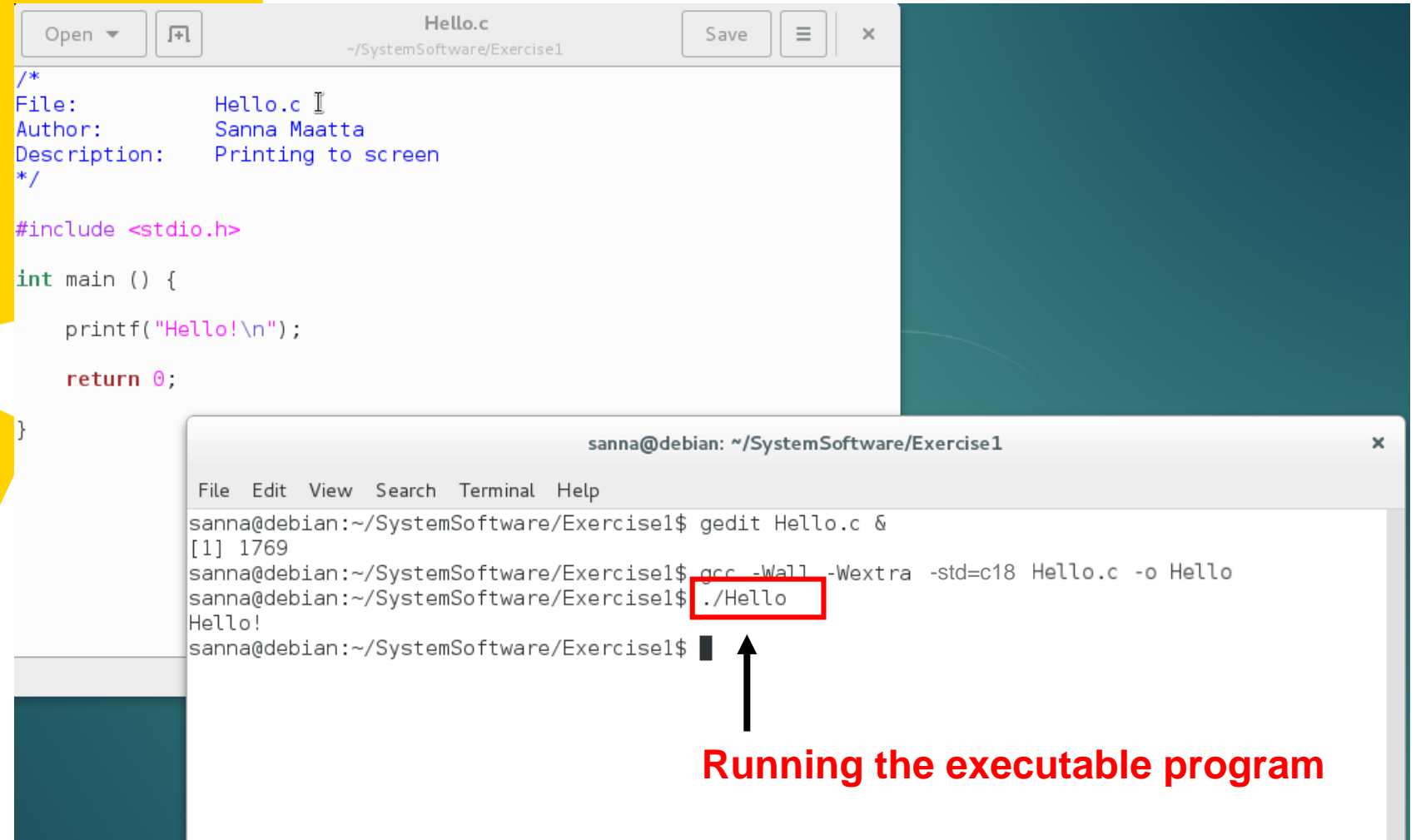
The screenshot displays a Linux desktop environment. In the background, a code editor window titled 'Hello.c' shows the source code for a simple C program. The code includes a comment block with file, author, and description information, followed by an include statement for `<stdio.h>` and a `main` function that prints 'Hello!' and returns 0. In the foreground, a terminal window titled 'sanna@debian: ~/SystemSoftware/Exercise1' shows the execution of the program. The user has opened the file with `gedit Hello.c &`, then compiled it using `gcc -Wall -Wextra -std=c18 Hello.c -o Hello` (this command is highlighted with a red box), and finally ran the executable with `./Hello`, which outputs 'Hello!'.

```
/*  
File:      Hello.c  
Author:    Sanna Maatta  
Description: Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
    return 0;  
}
```

```
sanna@debian: ~/SystemSoftware/Exercise1  
File Edit View Search Terminal Help  
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &  
[1] 1769  
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello  
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello  
Hello!  
sanna@debian:~/SystemSoftware/Exercise1$
```

-o makes an executable "Hello" from Hello.c

Compiling and Running on Linux



The screenshot displays a Linux desktop environment. In the background, a text editor window titled 'Hello.c' shows the source code of a C program. The code includes a comment block with file, author, and description information, followed by an include for `<stdio.h>` and a `main` function that prints 'Hello!' and returns 0. In the foreground, a terminal window titled 'sanna@debian: ~/SystemSoftware/Exercise1' shows the execution of the program. The user has run `gedit Hello.c &` to open the file in the editor. Then, they have run `gcc -Wall -Wextra -std=c18 Hello.c -o Hello` to compile the program. The command `./Hello` is shown being executed, with a red box highlighting it and an arrow pointing to it from the text below. The terminal output shows 'Hello!'.

```
/*  
File:      Hello.c  
Author:    Sanna Maatta  
Description: Printing to screen  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello!\n");  
    return 0;  
}
```

```
sanna@debian: ~/SystemSoftware/Exercise1  
File Edit View Search Terminal Help  
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &  
[1] 1769  
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello  
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello  
Hello!  
sanna@debian:~/SystemSoftware/Exercise1$
```

Running the executable program

**Notice, there is no file extension .c
any more in the executable's name**

Compiling and Running on Linux

```
/*
File:      Hello.c
Author:    Sanna Maatta
Description: Printing to screen
*/

#include <stdio.h>

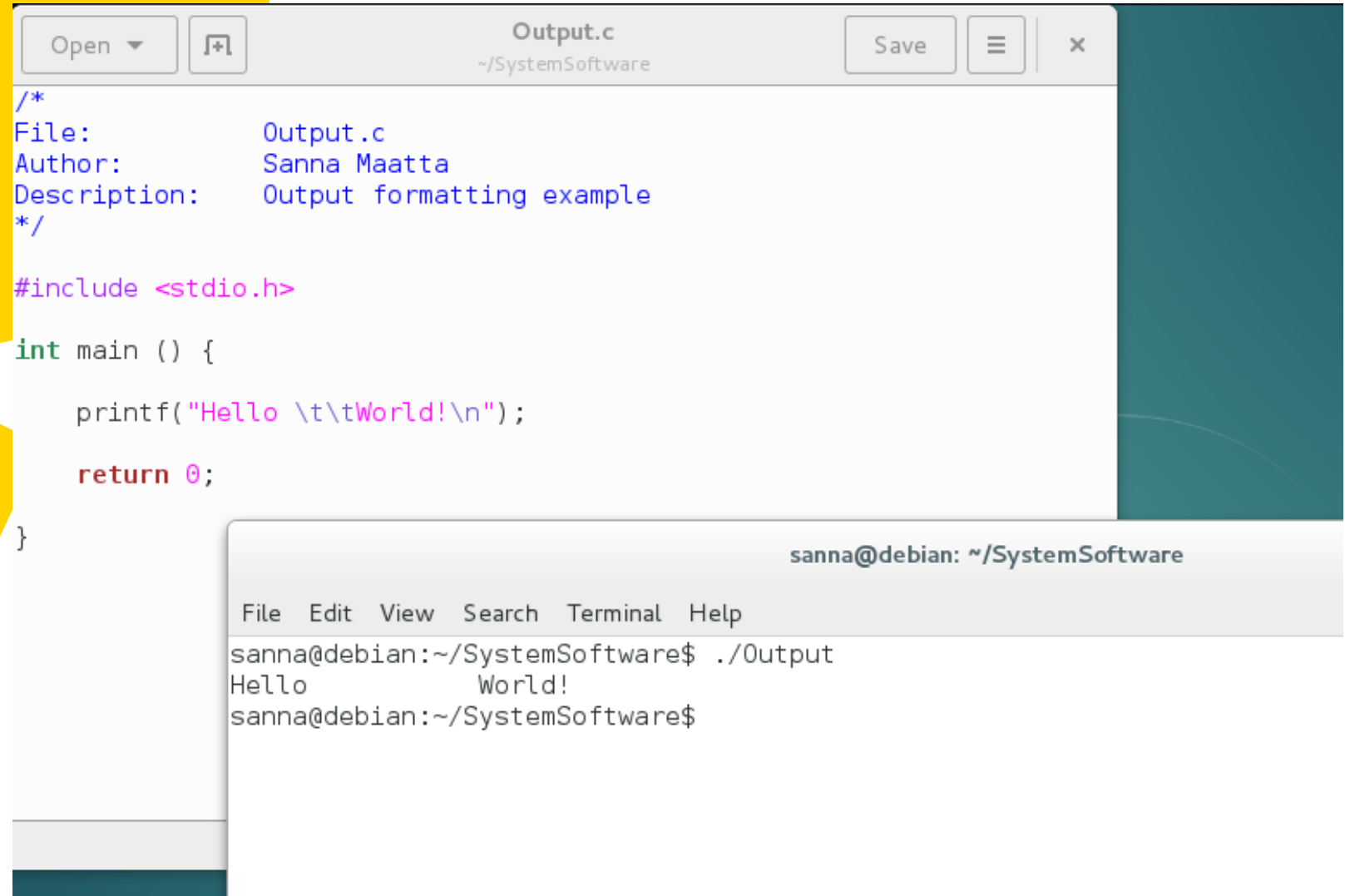
int main () {
    printf("Hello!\n");
    return 0;
}
```

```
sanna@debian: ~/SystemSoftware/Exercise1
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware/Exercise1$ gedit Hello.c &
[1] 1769
sanna@debian:~/SystemSoftware/Exercise1$ gcc -Wall -Wextra -std=c18 Hello.c -o Hello
sanna@debian:~/SystemSoftware/Exercise1$ ./Hello
Hello!
sanna@debian:~/SystemSoftware/Exercise1$
```

And this is the output print

Output Formatting in C

Output Formatting Example 1

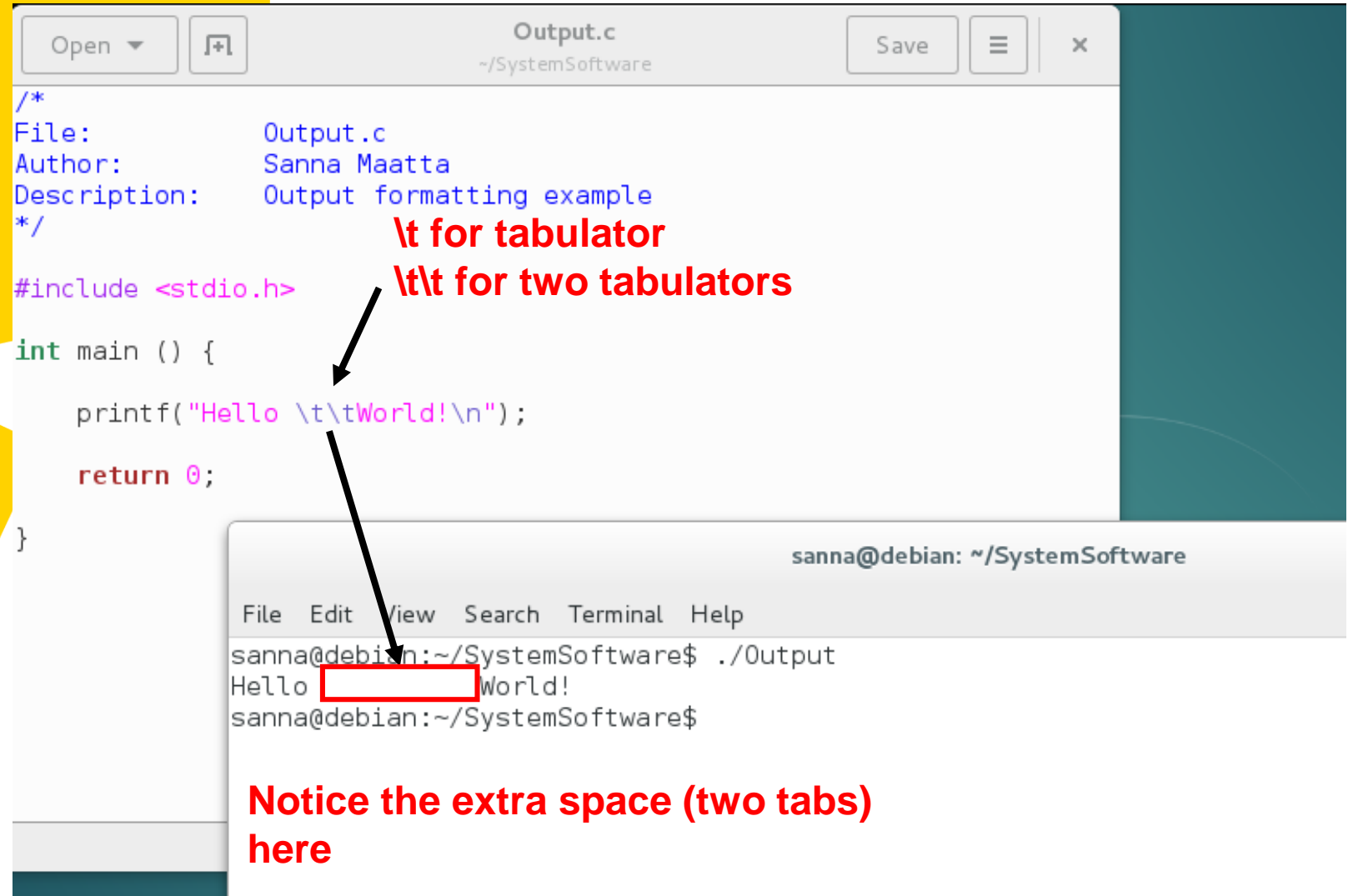


The image shows a code editor window titled "Output.c" with the file path "~/SystemSoftware". The code is a C program that prints "Hello World!". Below the code editor is a terminal window titled "sanna@debian: ~/SystemSoftware". The terminal shows the command `./Output` being executed, resulting in the output "Hello World!".

```
/*  
File:      Output.c  
Author:    Sanna Maatta  
Description: Output formatting example  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello \t\tWorld!\n");  
    return 0;  
}
```

```
sanna@debian: ~/SystemSoftware  
File Edit View Search Terminal Help  
sanna@debian:~/SystemSoftware$ ./Output  
Hello          World!  
sanna@debian:~/SystemSoftware$
```


Output Formatting Example 1



The image shows a code editor window titled "Output.c" with the following C code:

```
/*  
File:      Output.c  
Author:    Sanna Maatta  
Description: Output formatting example  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello \t\tWorld!\n");  
    return 0;  
}
```

Annotations in red text explain the escape sequences: `\t` for tabulator and `\t\t` for two tabulators. An arrow points from the `\t\t` in the code to the terminal output.

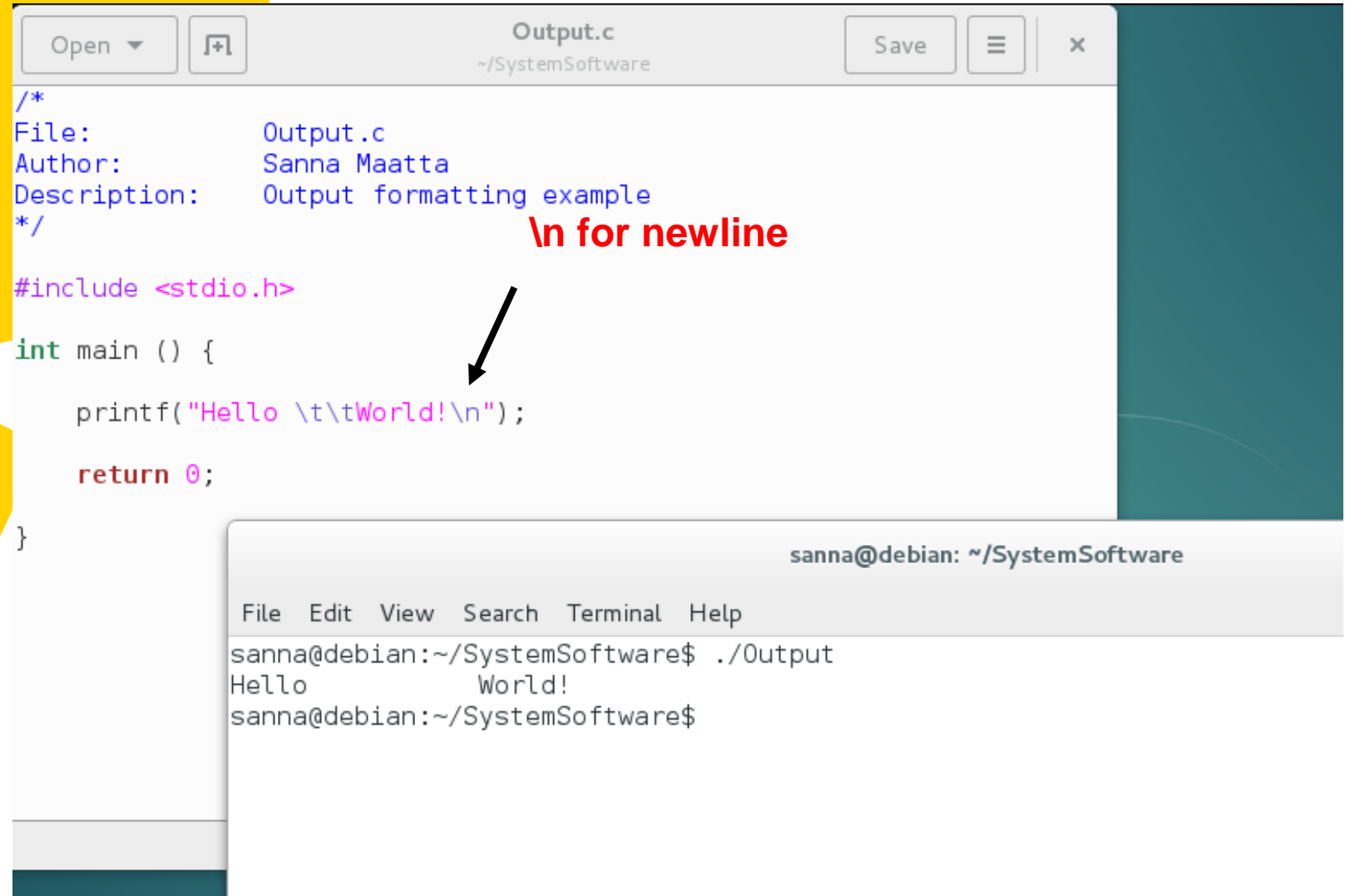
The terminal window shows the command `./Output` being executed, resulting in the output:

```
sanna@debian: ~/SystemSoftware  
File Edit View Search Terminal Help  
sanna@debian:~/SystemSoftware$ ./Output  
Hello   World!  
sanna@debian:~/SystemSoftware$
```

A red box highlights the extra space between "Hello" and "World!" in the terminal output. A red text annotation at the bottom states: "Notice the extra space (two tabs) here".

Output Formatting Example 1

- `\n` (newline)
- `\t` (tab)
- `\v` (vertical tab)
- `\f` (new page)
- `\b` (backspace)
- `\r` (carriage return)



The screenshot shows a code editor window titled "Output.c" with the following content:

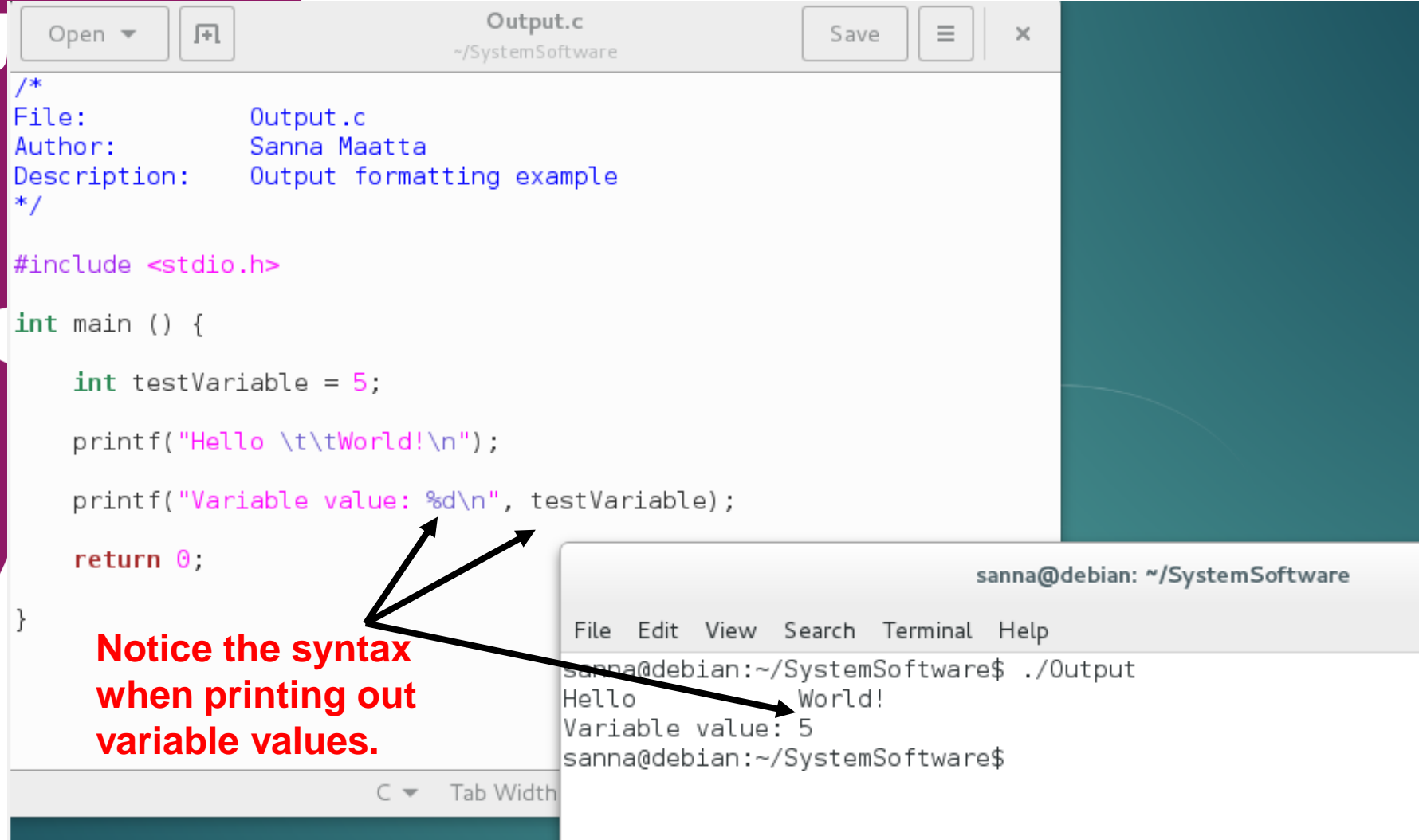
```
/*  
File:      Output.c  
Author:    Sanna Maatta  
Description: Output formatting example  
*/  
  
#include <stdio.h>  
  
int main () {  
    printf("Hello \t\tWorld!\n");  
    return 0;  
}
```

A red text label "`\n` for newline" with an arrow points to the `\n` escape sequence in the `printf` statement.

Below the code editor is a terminal window titled "sanna@debian: ~/SystemSoftware". It shows the command `./Output` being executed, resulting in the output:

```
sanna@debian:~/SystemSoftware$ ./Output  
Hello          World!  
sanna@debian:~/SystemSoftware$
```

Output Formatting Example 2



The image shows a code editor window titled "Output.c" with the following C code:

```
/*  
File:      Output.c  
Author:    Sanna Maatta  
Description: Output formatting example  
*/  
  
#include <stdio.h>  
  
int main () {  
    int testVariable = 5;  
    printf("Hello \t\tWorld!\n");  
    printf("Variable value: %d\n", testVariable);  
    return 0;  
}
```

Below the code editor is a terminal window titled "sanna@debian: ~/SystemSoftware". The terminal shows the command `./Output` being executed, resulting in the output:

```
sanna@debian:~/SystemSoftware$ ./Output  
Hello          World!  
Variable value: 5  
sanna@debian:~/SystemSoftware$
```

Three arrows point from the text "Notice the syntax when printing out variable values." to the `printf` statements in the code and the corresponding output in the terminal.

**Notice the syntax
when printing out
variable values.**

Output Formatting Example 2

```
Output.c
~/SystemSoftware

/*
File:      Output.c
Author:    Sanna Maatta
Description: Output formatting example
*/

#include <stdio.h>

int main () {

    int testVariable = 5;

    printf("Hello \t\tWorld!\n");

    printf("Variable value: %d\n", testVariable);

    return 0;

}
```

%d: integer
%c: character
%f: float and double
%x: hexadecimal

**Notice the syntax
when printing out
variable values.**

```
sanna@debian: ~/SystemSoftware
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello          World!
Variable value: 5
sanna@debian:~/SystemSoftware$
```

Output Formatting Example 2

```
Output.c
~/SystemSoftware

/*
File:      Output.c
Author:    Sanna Maatta
Description: Output formatting example
*/

#include <stdio.h>

int main () {
    int testVariable = 5;
    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d\n", testVariable);

    return 0;
}
```

**Always initialize
variables (style
guide rule 6).**

```
sanna@debian: ~/SystemSoftware
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello          World!
Variable value: 5
sanna@debian:~/SystemSoftware$
```

C Tab Width

Output Formatting Example 2

```
Output.c
~/SystemSoftware

/*
File:      Output.c
Author:    Sanna Maatta
Description: Output formatting example
*/

#include <stdio.h>

int main () {
    int testVariable = 5;
    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d\n", testVariable);
    return 0;
}
```

Always initialize
variables (style
guide rule 6).

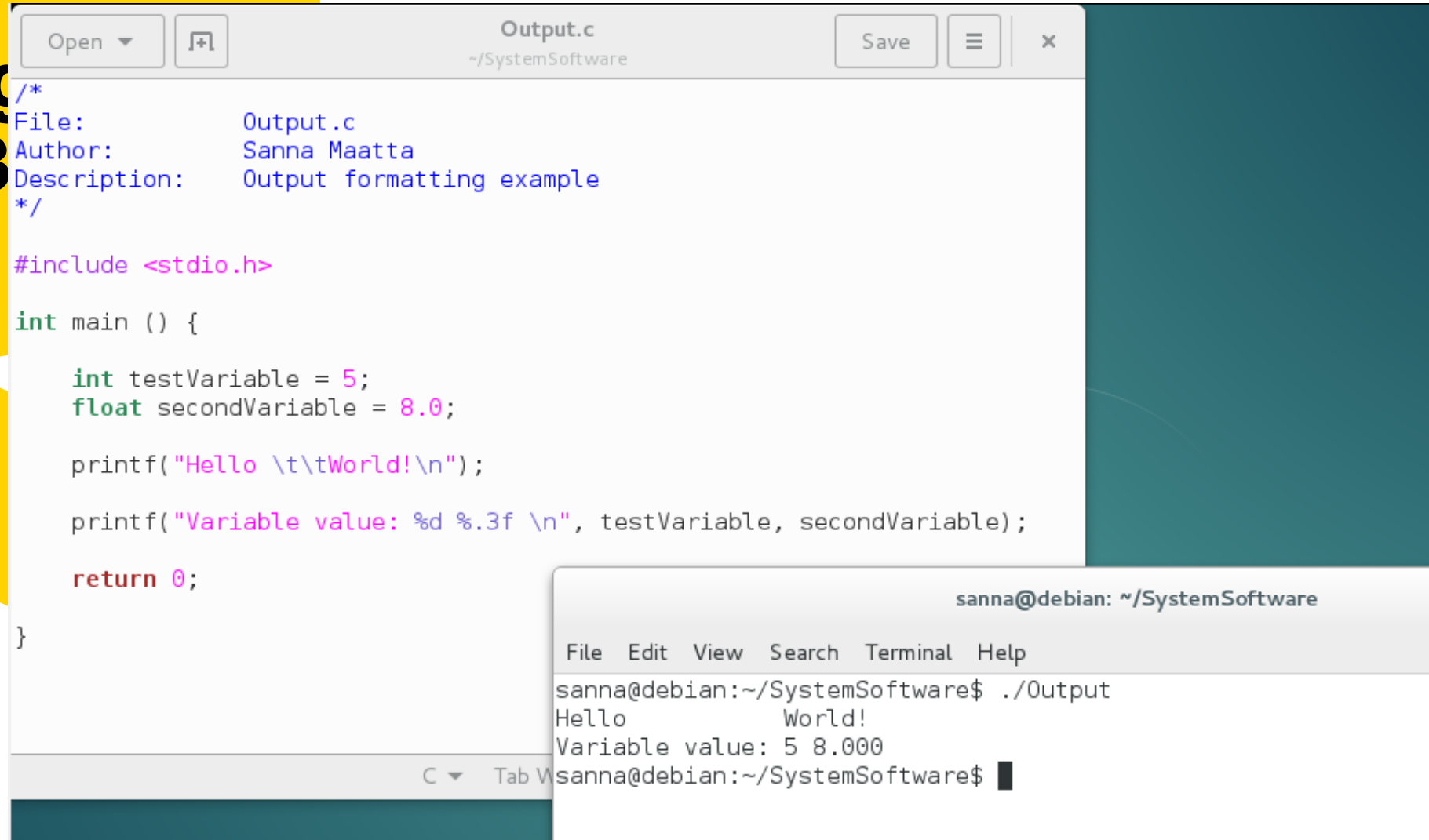
`int number;` // Un-initialized variable. Do not use!

`int number = 0;` // Initialized variable.

```
sanna@debian: ~/SystemSoftware

File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello          World!
Variable value: 5
sanna@debian:~/SystemSoftware$
```

Output Formatting Example 3



The image shows a code editor window titled "Output.c" with the following C code:

```
/*
File:      Output.c
Author:    Sanna Maatta
Description: Output formatting example
*/

#include <stdio.h>

int main () {

    int testVariable = 5;
    float secondVariable = 8.0;

    printf("Hello \t\tWorld!\n");

    printf("Variable value: %d %.3f \n", testVariable, secondVariable);

    return 0;
}
```

Below the code editor is a terminal window titled "sanna@debian: ~/SystemSoftware". The terminal shows the command `./Output` being executed, resulting in the output:

```
sanna@debian:~/SystemSoftware$ ./Output
Hello          World!
Variable value: 5 8.000
sanna@debian:~/SystemSoftware$
```

Output Formatting Example 3

```
/*
File:      Output.c
Author:    Sanna Maatta
Description: Output formatting example
*/

#include <stdio.h>

int main () {

    int testVariable = 5;
    float secondVariable = 8.0;

    printf("Hello \t\tWorld!\n");
    printf("Variable value: %d %.3f\n", testVariable, secondVariable);

    return 0;
}
```

Three decimal precision

```
sanna@debian: ~/SystemSoftware
File Edit View Search Terminal Help
sanna@debian:~/SystemSoftware$ ./Output
Hello          World!
Variable value: 5 8.000
sanna@debian:~/SystemSoftware$
```


Questions?

