

LAB00 TASK 2

ED ENVIRONMENT, CONFIGURATIONS, HELLO WORLD PROGRAM

BEESHANGA ABEWARDANA JAYAWICKRAMA

**UTS:
ENGINEERING AND
INFORMATION
TECHNOLOGY**

STEPS IN CREATING A C PROGRAM

When implementing a C program we go through 3 steps.

1. **Write** a C program
2. **Compile** the program with GCC
3. **Run** it on the terminal

We use **Ed** as the C programming environment. Click on the Ed link on UTSONline to access the Ed environment.

Following steps will guide you on how to **develop your first C program in Ed environment.**

WRITE – CREATE A NEW WORKSPACE

STEP 1: after logging into **Ed** selecting **48430 Fundamentals of C Programming** from the Dashboard, click on the **Workspaces** icon from the top menu bar.



STEP 2: click on the 'New Workspace' button and choose an appropriate workspace name.

The workspace name should have **alphanumeric characters and underscores ONLY** (NO spaces).

Suggested workspace name 48430.

Click on the created workspace to access it.

WRITE – CONFIGURE THE EDITOR

STEP 3: turn on the following configurations in the **Settings**:

- Line numbers
- Detect Indentation
- Smart Indent

STEP 4: Set Tab Size to 4

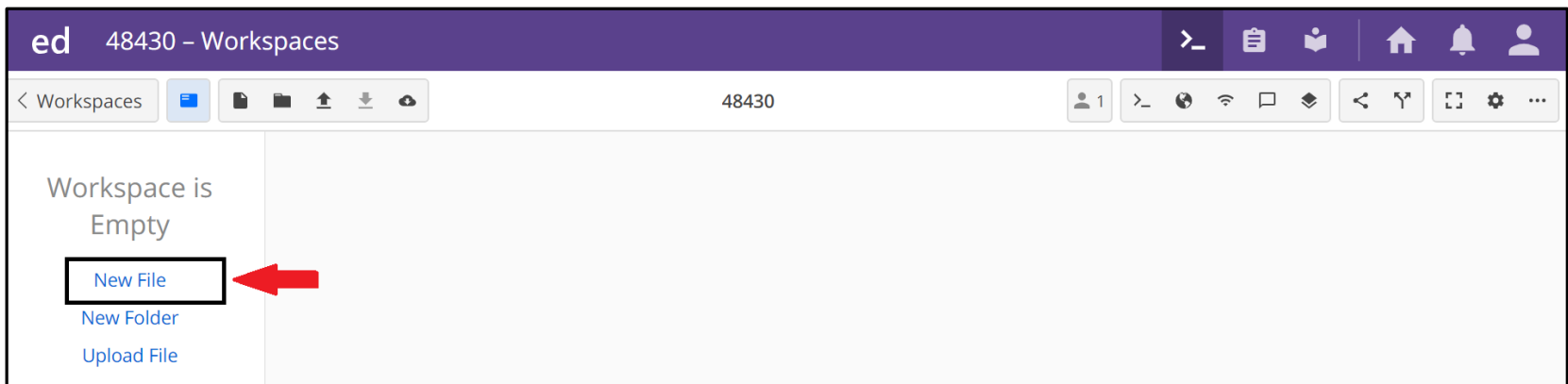
Settings	
Settings will be stored on this device only.	
<input type="checkbox"/>	Word Wrap Automatically break long lines.
<input checked="" type="checkbox"/>	Line Numbers Show line numbers in editor gutter.
<input type="checkbox"/>	Highlight Active Line Highlight the line containing the cursor.
<input type="checkbox"/>	Show Invisibles Show invisible (whitespace) characters.
<input checked="" type="checkbox"/>	Detect Indentation Detect soft tabs and tab size from editor initial content.
<input type="checkbox"/>	Soft Tabs Insert spaces instead of tab characters.
<input checked="" type="checkbox"/>	Smart Indent Use context-sensitive indentation.

WRITE – CREATE A NEW C FILE

STEP 5: Click on '**New File**' from left hand side menu pane. Select an appropriate file name.

Suggested filename is `helloworld.c` and **.c in the end is important.**

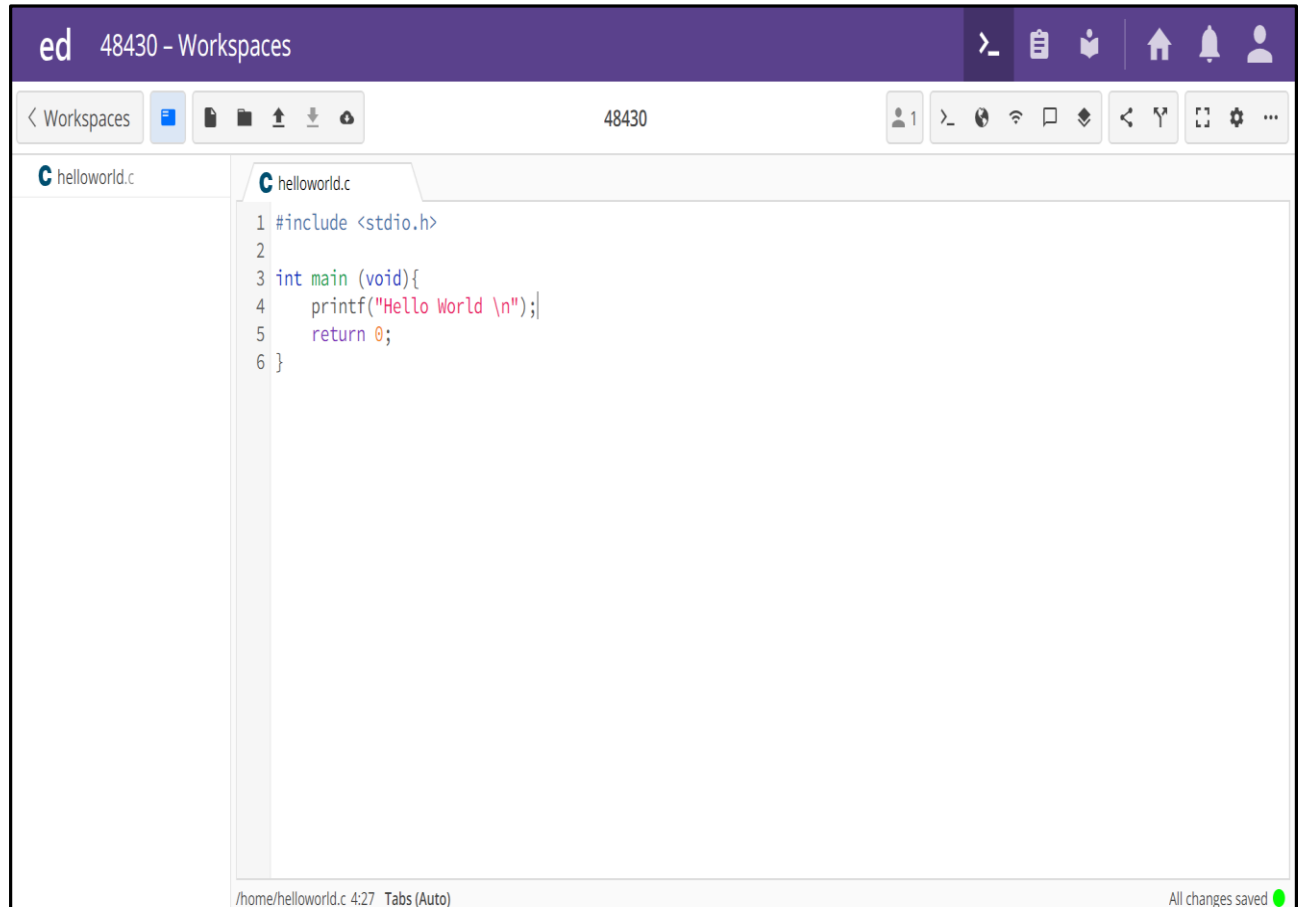
The filename should have **alphanumeric characters and underscores ONLY** (NO spaces).



WRITE – CREATE A NEW C FILE

STEP 6: type in the following C program.

Manually enter the program (no copy paste).



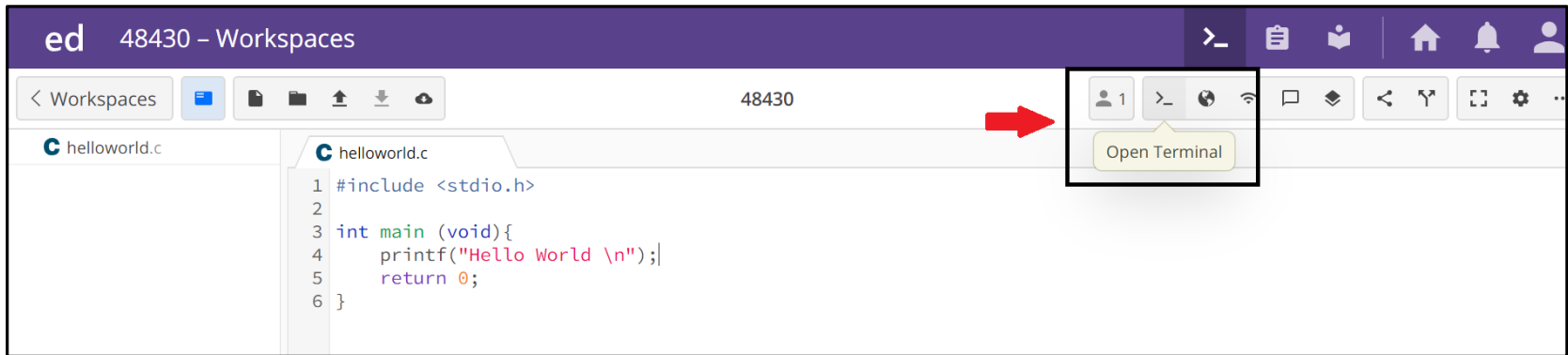
The screenshot shows the ed editor interface. The top bar is purple with the text 'ed 48430 - Workspaces' and various icons. Below the bar, there's a toolbar with icons for file operations. The main editing area shows a file named 'helloworld.c' with the following C code:

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

The status bar at the bottom indicates the file path as '/home/helloworld.c 4:27 Tabs (Auto)' and shows 'All changes saved' with a green dot.

COMPILE – OPEN A TERMINAL

STEP 7: open a new terminal.



COMPILE – COMPILE AND LINK

STEP 8 Compile and link the program as follows

```
gcc -Wall -Werror -ansi -o helloworld.out helloworld.c -lm
```

If you typed the program correctly you should NOT see any error messages. If there are any error messages go to step 6.

```
>_ user@sahara:~  
[user@sahara ~]$ gcc -Wall -Werror -ansi -o helloworld.out helloworld.c -lm  
[user@sahara ~]$ █
```


RUN

STEP 9: Run the program as follows.

```
./helloworld.out
```

The **./** in front is important.

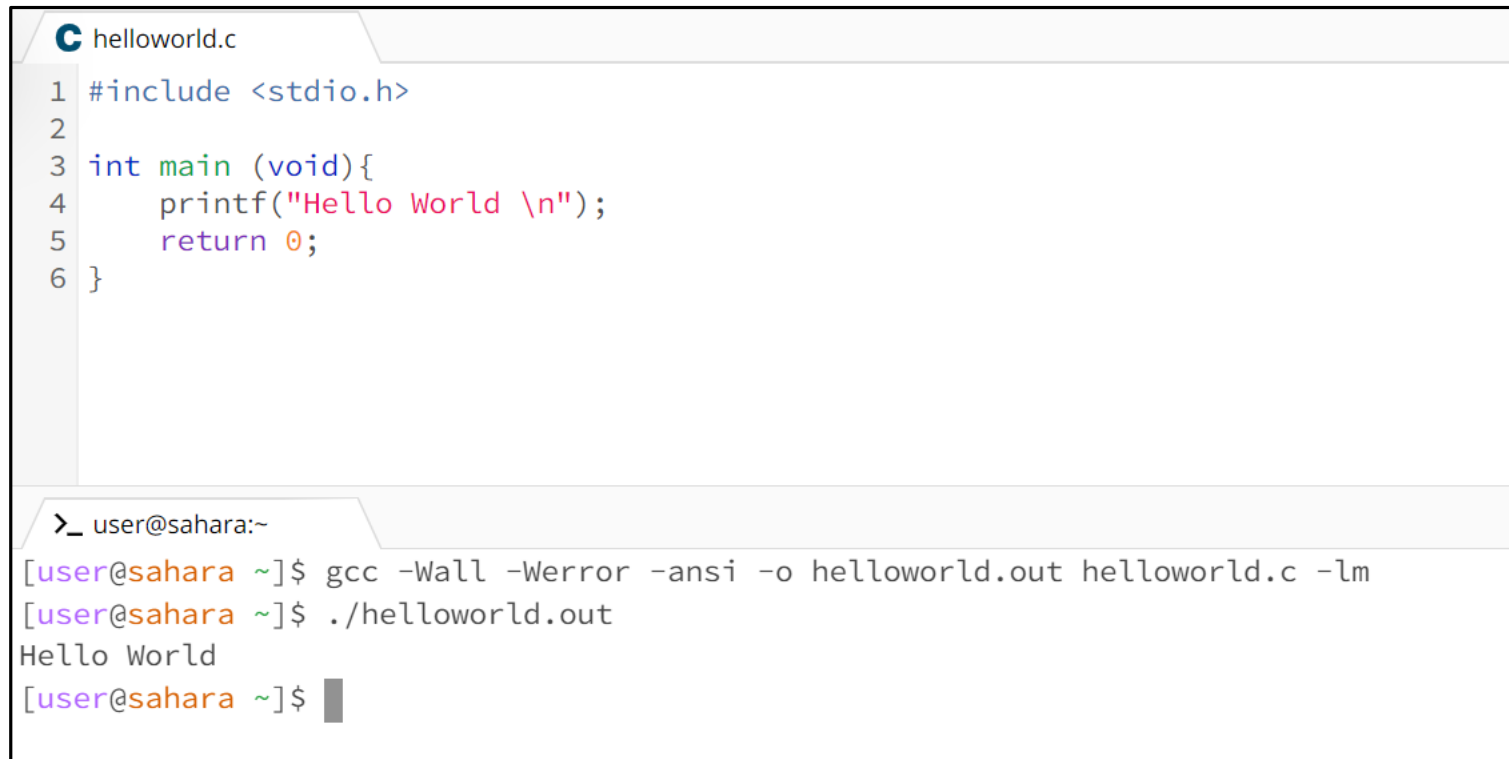
If the program runs successfully, it should print "Hello World"

```
>_ user@sahara:~  
[user@sahara ~]$ gcc -Wall -Werror -ansi -o helloworld.out helloworld.c -lm  
[user@sahara ~]$ ./helloworld.out  
Hello World  
[user@sahara ~]$ █
```

PLAY TIME

While you have both text editor and terminal open, go and change the helloworld.c file to print a different message, come back to the terminal to compile and run it.

You only have to repeat Step 6, 8 and 9 (no need to close and reopen the .c file/terminal every time you make a change).



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

>_ user@sahara:~
[user@sahara ~]$ gcc -Wall -Werror -ansi -o helloworld.out helloworld.c -lm
[user@sahara ~]$ ./helloworld.out
Hello World
[user@sahara ~]$
```

EXERCISE I

Research more about the command we typed to compile and link the source code.

```
gcc -Wall -Werror -ansi -o helloworld.out helloworld.c -lm
```

- What do the above terms mean?
- What happens if you type:

```
gcc -Wall -Werror -ansi helloworld.c -lm
```

- Why do we use:

```
-Wall -Werror -ansi    and    -lm
```

- Write your own program to do further testing.

EXERCISE II

1. Understand the significance of each line in the helloworld program given in Step 6 (do research, read the textbook).
2. Understand the significance of each term in compilation syntax in Step 8 (do research, read the textbook).
3. Experiment. Change the program to print your name in a separate line after "Hello World" (do research, read the textbook, experiment).

ADVANCED WORKSPACE FUNCTIONALITIES

You can share your workspace with your friends to work on a programme together.

This will be come when you work on the group project later in the semester.

Read about 'Sharing' and 'Forking' workspaces and give it a try.

