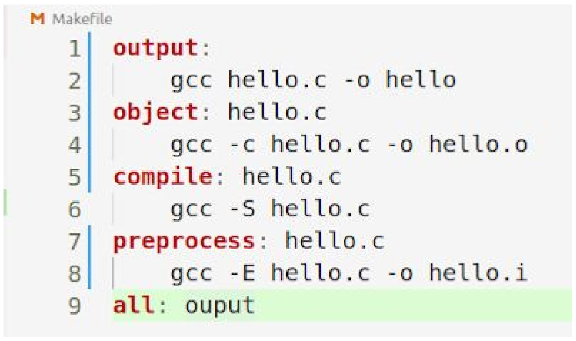
Tushar Mohan
2019393

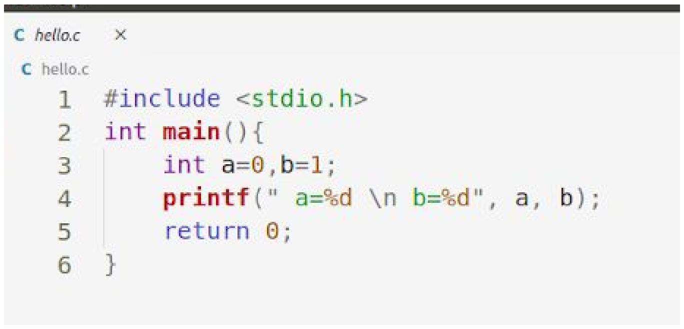
C compilation Steps Write-up

- **Makefile**



```
output:
	gcc hello.c -o hello
object: hello.c
	gcc -c hello.c -o hello.o
compile: hello.c
	gcc -S hello.c
preprocess: hello.c
	gcc -E hello.c -o hello.i
all: ouput
```

- **C program**



```
#include <stdio.h>
int main(){
    int a=0,b=1;
    printf(" a=%d \n b=%d", a, b);
    return 0;
}
```

1. **Preprocessing step:**

This step includes preprocessing directives mentioned in the code written in C language which are the include lines, typedefs, the define, etc. It also removes all the comments from the code.

Command used to do the above mentioned step:

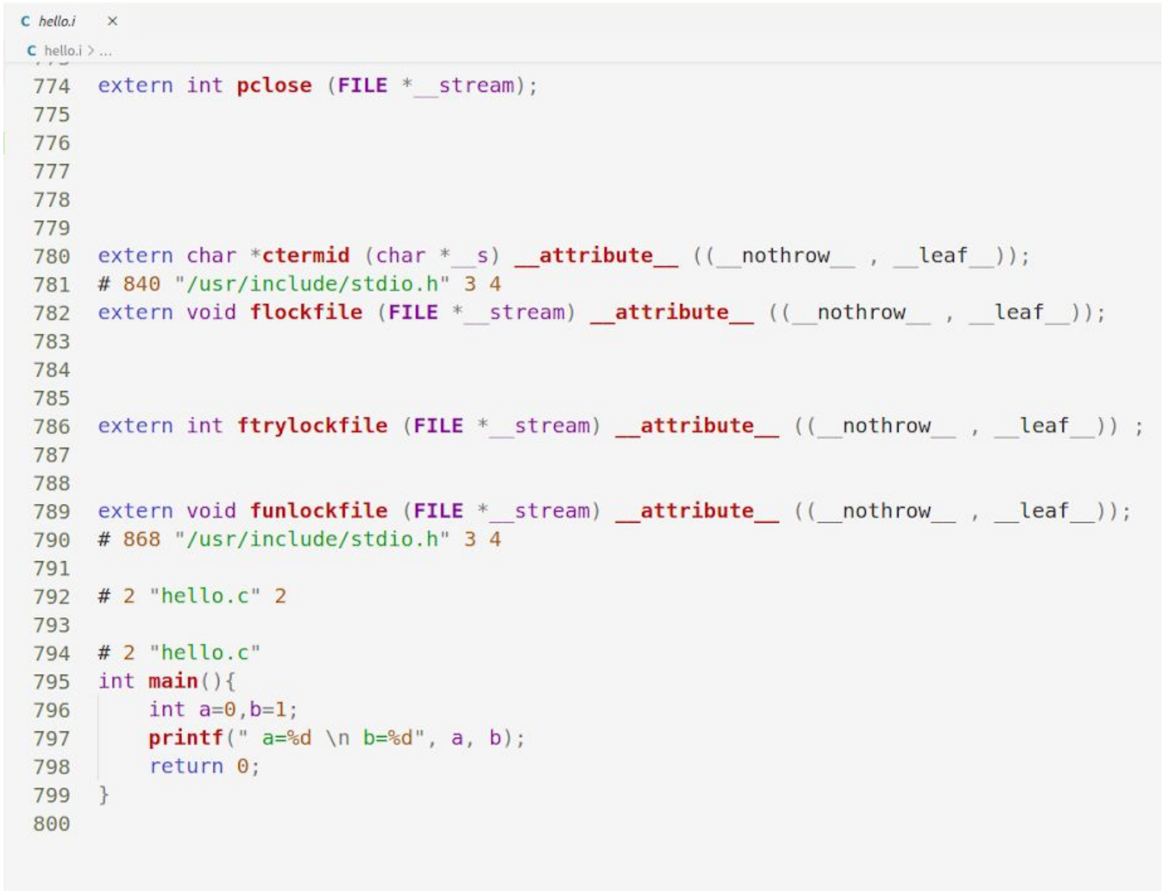
gcc -E hello.c -o hello.i

Makefile command for the same :

make preprocess

Result of the above command is preprocessed file named “hello.i” which is readable and itself written in C language, but the difference between this and the original C program is that all the preprocessing has been done and it is free from any comments that might be present in the C code. This file includes various new lines introduced by the command we used above.

- **An excerpt from the hello.i file**



```
extern int pclose (FILE *__stream);





extern char *ctermid (char *__s) __attribute__ ((__nothrow__ , __leaf__));
# 840 "/usr/include/stdio.h" 3 4
extern void flockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__));



extern int ftrylockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__)) ;


extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__));
# 868 "/usr/include/stdio.h" 3 4

# 2 "hello.c" 2

# 2 "hello.c"
int main(){
    int a=0,b=1;
    printf(" a=%d \n b=%d", a, b);
    return 0;
}

```

2. **Compilation step:**

This step includes compiling the code written in a high level language. It essentially converts the high level code to assembly language code which will later be converted to machine language code.

Command used to do the above mentioned step:

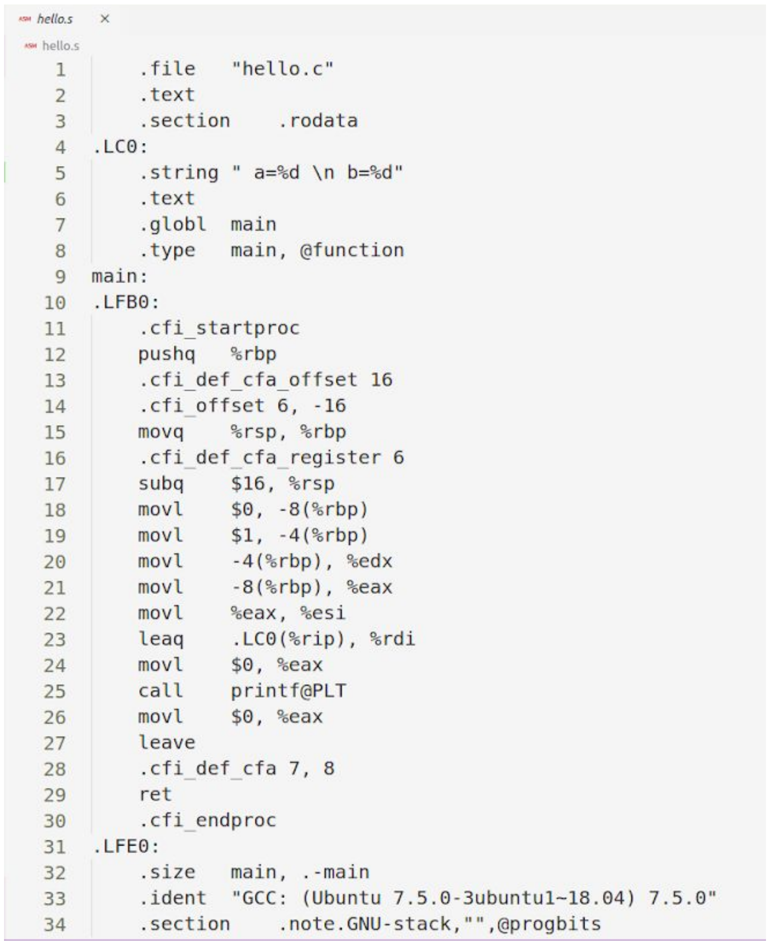
```
gcc -S hello.c
```

Makefile command for the same:

```
make compile
```

Result of the above command is a file named "hello.s" which if viewed will show a code written in assembly language using mnemonics. It has all the dependencies intact and gives an insight as to what is happening at an assembly level.

- **hello.s file :**



```
	.file	"hello.c"
	.text
	.section	.rodata
.LC0:
	.string " a=%d \n b=%d"
	.text
	.globl	main
	.type	main, @function
main:
.LFB0:
	.cfi_startproc
	pushq	%rbp
	.cfi_def_cfa_offset 16
	.cfi_offset 6, -16
	movq	%rsp, %rbp
	.cfi_def_cfa_register 6
	subq	$16, %rsp
	movl	$0, -8(%rbp)
	movl	$1, -4(%rbp)
	movl	-4(%rbp), %edx
	movl	-8(%rbp), %eax
	movl	%eax, %esi
	leaq	.LC0(%rip), %rdi
	movl	$0, %eax
	call	printf@PLT
	movl	$0, %eax
	leave
	.cfi_def_cfa 7, 8
	ret
	.cfi_endproc
.LFE0:
	.size	main, .-main
	.ident	"GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
	.section	.note.GNU-stack,"",@progbits
```

3. **Assembly step:**

This step includes converting the assembly code into machine language code, i.e., object code. This file is non-readable by a human being.

Command used to do the above mentioned step:

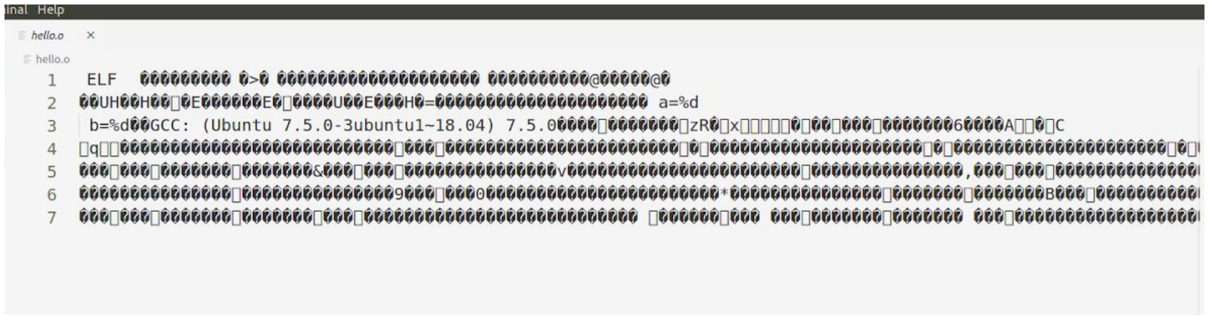
gcc -c hello.c -o hello.o

Makefile command for the same:

make object

Result of the above command is a file named “hello.o” which essentially the binary code. The reason we don’t see the readable version in the hello.o is because after this it has to be read by the circuitry of the machine in this particular way.

- **hello.o file as available to read:**



```
 ELF  ���������� �>� ����������������������� �����������@������@�
��UH��H��▯�E�������E�▯�����U��E���H�=������������������������ a=%d
 b=%d��GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0�����▯��������▯zR�▯x▯▯▯▯▯�▯��▯���▯��������6�����A▯▯�▯C
▯q▯▯�����������������������������▯���▯���������������������������▯�▯�������������������������▯�▯�������������������������▯�▯
���▯���▯��������▯��������&���▯���▯�����������������v��������������������������▯�����������������,���▯���▯����������������
����������������▯������������������9���▯���0��������������������������*������������������▯��������▯��������B���▯�����������
���▯���▯��������▯��������▯���▯��������������������������������� ▯�������▯��� ���▯���������▯�������� ���▯����������������������
```

4. **Linking step:**

This step includes taking into account all the files on which the C code depends on. In this case, it doesn't depend on any other C code written, but if a code does depend on functions from some other codes as well then this step is essential and on top of that, this code results in an executable file which can then be run to see the output of the program.

Command used to do the above mentioned step:

```
gcc hello.c -o hello
```

Makefile command for the same:

```
make
```

Result is hello file which can then be executed with command: ./hello (if in the same directory).

When only the last command is run then also all the previous steps take place but the files of the previous steps are not made. The reason is due to the cleanup protocols of the gcc command that are used in case no flag like -E, -S, etc. are used.

- **An excerpt of hello file as available to be read:**

hello ×
hello
The Marketplace has extensions that can help with '.
Search Marketplace
Don't Show
The Marketplace has extensions that can help with '.
Search Marketplace
Don't Show

```
 ELF  ��������� �>� ���@ ������@��������0□�����������@�8�    �@�□�□� ��� ���@�������@�������@�������� ������� ����
�������
 ������
 �����X□������`□��������� �����□���□����
�������
 ������
 ������□�������□������□�������□���□���T□������T□������T□������D�������D�������□�������P�td□���□□������□□������
�������
 ������
 �����H□������H□������□�������/lib64/ld-linux-x86-64.so.2�□���□���□���GNU�����□���□�������□���□���□���GNU�□piT
 �����□�������@□�������
 �����□���������□������□□ �����□��������□□ ������□ �����□���□������������□ �����□���□������������□ �����□���□����
 �H��t□��H��□������������5�
 ��%�
 �□□@��%�
 �h�����������%�
 �f����������1�I��^H��H���PTL�□�□��H�
#□��H�=�����□v
 ��□□D��H�=�
 �UH�□�
 �H9�H��t□H�□J
 �H��t
]��f.□□�������]�□□@�f.□□�������H�=Y
 �H�5R
 �UH)�H��H��□H��H��?H□�H��t□H�□□
 �H��t]��f□□������]�□□@�f.□□��������=
 ��u/H�=�    ��UH��tH�=�    ��
����H�����□�  �□]�□□��������f□□D��UH��]�f���UH��H��□�E�������E�□����U��E���H�=��������
 b=%d����□□□;8����□������������□��������,���T���6��������l�����������,□��□��������□zR�□x
□q□□���D���|��������e����B□□�□B□□�□E□ �□B□(�□H□0�□H□8�□M□@r□8A□0A□(B□ B□□B□□B□□�□����
��������□�������□���������
 �����□��������□��������□���������
 �����□��������□�����������o�����□�������□��������`□�������□���������□�������
����������������□��������□��������□����������������□���������□ �����□��������□��������□����
```

References:

https://www.cprogramming.com/compilingandlinking.html