Gcc command-line options to pause compilation at each step.

- The first command-line option to pause at the pre-processing step is **gcc -E hello.c**
  This command stops the compilation right after pre-processing and the output of this step is stored in a file with a ".i" extension that is hello.i.

- The second command-line option to pause at the compilation step is **gcc -S hello.c**
  This command takes in hello.i and generates the intermediate compiled output which is actually the assembly code. So this will create a file with ".s" extension hello.s.

- To see the output by the assembler we use **gcc -c hello.c**
  It will take the .s file and convert it into the object code, the machine language code. This will produce a file with ".o" extension hello.o.

- After this, to see the output of the last step where an executable file is created, we use **gcc -o hello hello.c** where hello is the executable file.
  This is where the linker comes into play, it will take in .o file and produce the executable file.

Description of the outcomes of each step involving the description of the output file.

1. **The Preprocessor**
   After this .i file is created. The output does not contain any comments added by the user, it includes the code of the header file stdio.h and the macros are replaced by their values.
   The end of the output looks like this:

```
extern void flockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__));



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


extern void funlockfile (FILE *__stream) __attribute__ ((__nothrow__ , __leaf__));
# 858 "/usr/include/stdio.h" 3 4
extern int __uflow (FILE *);
extern int __overflow (FILE *, int);
# 873 "/usr/include/stdio.h" 3 4

# 2 "hello.c" 2


# 3 "hello.c"
int main(){
    int a = 10;
    int b = 20;
    printf("%d %d\n", a, b);
    return 0;
}
```

2. **The Compiler**

   In this step the .i file is compiled and an intermediate compiled output is generated which are actually the assembly level instructions. The output can be seen on any text editor.

   The output looks like this:

```
    .text
    .section    .rodata
.LC0:
    .string "%d %d\n"
    .text
    .globl  main
    .type   main, @function
main:
.LFB0:
    .cfi_startproc
    endbr64
    pushq   %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    subq    $16, %rsp
    movl    $10, -8(%rbp)
    movl    $20, -4(%rbp)
    movl    -4(%rbp), %edx
    movl    -8(%rbp), %eax
    movl    %eax, %esi
    leaq    .LC0(%rip), %rdi
```

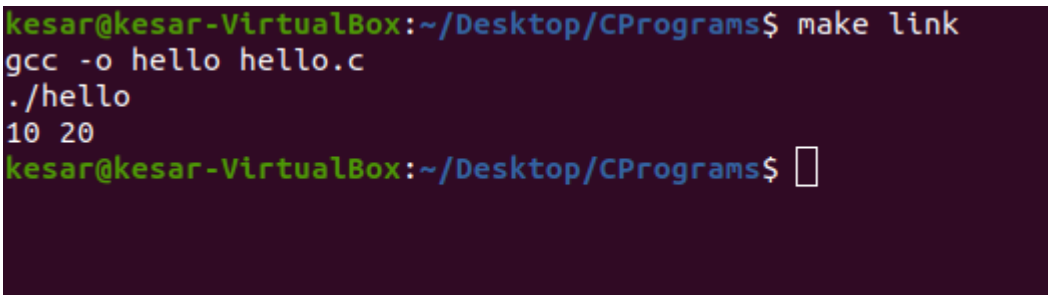
3. **The Assembler**

   The output looks like this:

```
7f45 4c46 0201 0100 0000 0000 0000 0000
0100 3e00 0100 0000 0000 0000 0000 0000
0000 0000 0000 0000 2803 0000 0000 0000
0000 0000 4000 0000 0000 4000 0e00 0d00
f30f 1efa 5548 89e5 4883 ec10 c745 f80a
0000 00c7 45fc 1400 0000 8b55 fc8b 45f8
89c6 488d 3d00 0000 00b8 0000 0000 e800
0000 00b8 0000 0000 c9c3 2564 2025 640a
0000 4743 433a 2028 5562 756e 7475 2039
2e33 2e30 2d31 3075 6275 6e74 7532 2920
392e 332e 3000 0000 0400 0000 1000 0000
0500 0000 474e 5500 0200 00c0 0400 0000
0300 0000 0000 0000 1400 0000 0000 0000
017a 5200 0178 1001 1b0c 0708 9001 0000
1c00 0000 1c00 0000 0000 0000 3a00 0000
0045 0e10 8602 430d 0671 0c07 0800 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0100 0000 0400 f1ff
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0300 0100 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0300 0300
0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0300 0400 0000 0000 0000 0000
```

The assembler transforms the intermediate compiled output into the machine language code which cannot be understood as it is not human readable. It is the object code which can be understood by the machine.

4. **The Linker**
   It creates the final executable file. It basically links the source files together by linking their object codes. It also links the function calls with their definition from the library file and if name not given, a.out is created. Here the name of the executable file is hello.



Makefile Command Description

1. To pause at the pre-process step, the target used is called ‘preprocess’. The user needs to type “**make preprocess**” to see the output.
2. The user must type “**make compile**” to pause after the compilation.
3. To see the .o file “**make assemble**” must be typed in the shell.
4. The final executable file can be run by typing in “**make link**”.
5. To clear all the extra files created while pausing at each step user must type in “**make clean**”.