

1. gcc program.c ^{Output} ⇒ a.out (Binary/Executable)
2. gcc program.c -o program.out ⇒ program.out (Binary/Executable)

gcc - compiler

program.c → source file

a.out / program.out → output file produced by compiler

Background Process or Compilation process -

→ program.c ⇒ This is source file

→ preprocessing ⇒ It's done by a compiler's unit "PREPROCESSOR"

↓

→ .Compilation ⇒ #include #define #ifdef ⇒ #include <stdio.h>

↓

→ Assembly ⇒ .S → Assembled → .o (Object file)

↓

→ .Linking ⇒ .o file + definition or code of printf()

↓

→ Executable file (a.out / program.out) ⇒ .o → .out (executable)

```
#include <stdio.h> 100 lines
int main()           - int main
{                     - {
    extension .c     - } 104
```

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

→ ./a.out → Running on command line

Kernel (OS) ⇒ 1. It make sure executable is in ELF format
2. check for its architecture (32 bit / 64 bit)
3. checks Headers in executable

↓

→ Virtual Memory layout creation (virtual address space)

↓

→ This binary is mapped to

↓

Dynamic linker is loaded. (ldd)

↓

Kernel creates thread -

↓

execution starts from -start

↓

Runtime Initialization

↓

preparation of argc, argv, environment variables

↓

main() // execution starts

a.out (executable)

↳ printf() → a library function .lib .so

==

The complete life cycle of a C program.

```
graph TD
    A[program.c] --> B[gcc]
    B --> C[Executable (a.out or something.out or .bin)]
```

