1.- Proceso de compilación en C

La compilación de un programa C se realiza en varias fases que normalmente son automatizadas y ocultadas por los entornos de desarrollo:

1.- Preprocesado: consistente en modificar el código fuente en C según una serie de instrucciones (denominadas directivas de preprocesado) simplificando de esta forma el trabajo del compilador.

Las directivas de preprocesado van precedidas por el símbolo #

a) **`#include <stdio.h>`** sustituirá esta línea por el contenido del archivo ***`stdio.h`***

b) **`#define c1 50`** sustituirá todas las apariciones de ***c1*** por ***50*** en cada aparición en el programa

c) Compilación condicional con las directivas ***`#ifdef, #ifndef, #else, #elif y #endif`***

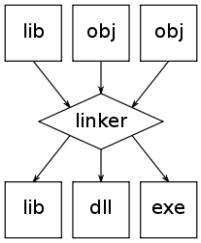
d) Eliminación de comentarios, ...

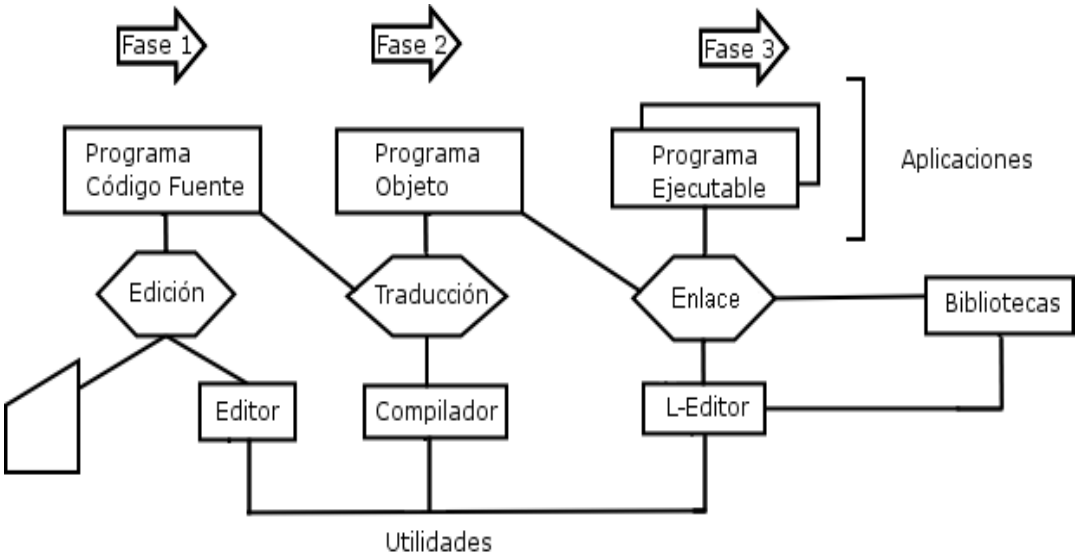
2.- Compilación que genera el código objeto a partir del código ya preprocesado.

Genera un archivo de código objeto por cada archivo de código fuente.

`Programa1.c -> Programa.obj`

3.- Enlazado que une los códigos objeto de los distintos módulos y bibliotecas externas (como las bibliotecas del sistema) para generar el programa ejecutable final.





2.- DLL: Dynamic Link Library = Biblioteca de Enlace Dinámico

Son archivos con código ejecutable que son invocados por otros programas.

En vez de incluir ese código ejecutable en el programa .exe que lo usa, se ha dejado en otro archivo .dll

Ventajas:

- **Reducen el tamaño de los archivos ejecutables:** Gran parte del código puede estar almacenado en bibliotecas y no en el propio ejecutable lo que redunda en una mejor modularización
- **Pueden estar compartidas entre varias aplicaciones:** Si el código es suficientemente genérico, puede resultar de utilidad para múltiples aplicaciones (por ejemplo, la MFC es una biblioteca dinámica con clases genéricas que recubren la API gráfica de Windows y que usan gran parte de las aplicaciones).
- **Facilitan la gestión y aprovechamiento de la memoria del sistema:** La carga dinámica permite al sistema operativo aplicar algoritmos que mejoren el rendimiento del sistema cuando se carguen estas bibliotecas. Además, al estar compartidas, basta con mantener una copia en memoria para todos los programas que la utilicen.
- **Brindan mayor flexibilidad frente a cambios:** Es posible mejorar el rendimiento o solucionar pequeños errores distribuyendo únicamente una nueva versión de la biblioteca dinámica. Nuevamente, está corrección o mejora será aprovechada por todas las aplicaciones que compartan la biblioteca.

Inconvenientes

- Que la instalación de un programa reemplace una DLL con una nueva versión incompatible
- Que la desinstalación del programa borre una DLL compartida