

# Tutorial Avanzado sobre el Lenguaje C

Bienvenidos a este tutorial avanzado sobre el lenguaje C. Este curso está diseñado para programadores y estudiantes de informática que desean profundizar sus conocimientos en C, explorando desde elementos obsoletos pero aún válidos hasta técnicas avanzadas de programación de red. A lo largo de este tutorial, examinaremos aspectos críticos del lenguaje, incluyendo nuevas palabras clave, convenciones de llamada, manejo de memoria, y programación concurrente. Prepárate para sumergirte en los detalles técnicos y las mejores prácticas que harán de ti un programador de C más competente y efectivo.

 by Angel Coyoy





# Elementos Obsoletos pero Válidos en C

1

## **gets() y sus Alternativas**

La función `gets()`, eliminada en C11, era utilizada para la entrada de cadenas. Sin embargo, su uso puede llevar a desbordamientos de búfer. Actualmente, se recomienda usar `fgets()` o la más segura `gets_s()` introducida en C11.

2

## **Funciones de Manipulación de Cadenas**

`strcpy()`, `strcat()`, y `sprintf()` son consideradas inseguras debido a posibles desbordamientos. Las alternativas seguras son `strncpy()`, `strncat()`, y `snprintf()`, que permiten especificar límites de tamaño.

3

## **Operadores de Asignación Compuesta**

Aunque no obsoletos, operadores como `+=` y `-=` a veces se evitan en código crítico para seguridad. Algunos programadores prefieren usar asignaciones explícitas para mayor claridad y control.

# Nuevas Palabras Clave en C Moderno

1

## **\_Bool para Lógica Booleana**

Introducido en C99, `_Bool` proporciona un tipo de dato booleano nativo. Se puede usar con las macros `true` y `false` definidas en `<stdbool.h>`. Esto mejora la legibilidad del código al trabajar con condiciones lógicas.

2

## **inline para Optimización**

La palabra clave `inline` sugiere al compilador que expanda una función en el lugar de su llamada. Esto puede mejorar el rendimiento al eliminar la sobrecarga de las llamadas a funciones, especialmente útil para funciones pequeñas y frecuentemente utilizadas.

3

## **`_Complex` y `_Imaginary`**

Estas palabras clave soportan operaciones con números complejos. `_Complex` permite declarar variables de tipo complejo, mientras que `_Imaginary` (aunque raramente implementado) está destinado a representar la parte imaginaria de un número complejo.



# Convenciones de Llamada en C

## cdecl (C Declaration)

Esta convención es la estándar en C. Los argumentos se pasan de derecha a izquierda en la pila, y el llamador es responsable de limpiar la pila después de la llamada. Es flexible pero puede ser menos eficiente para funciones con muchos argumentos.

## stdcall

Utilizada principalmente en la API de Windows, stdcall pasa los argumentos de derecha a izquierda, pero la función llamada limpia la pila. Esto resulta en un código más compacto cuando una función se llama frecuentemente, ya que la limpieza de la pila se codifica una sola vez en la función.

## fastcall

Esta convención pasa algunos argumentos en registros en lugar de la pila, lo que puede acelerar las llamadas a funciones. La implementación exacta varía según el compilador y la arquitectura, pero generalmente los primeros dos o tres argumentos se pasan en registros.

# Pilas y Recursividad en C

1

## Estructura de la Pila de Llamadas

La pila de llamadas es una estructura LIFO que almacena información sobre las funciones activas. Cada vez que se llama a una función, se crea un nuevo marco de pila que contiene variables locales, parámetros y la dirección de retorno.

2

## Recursividad y su Impacto en la Pila

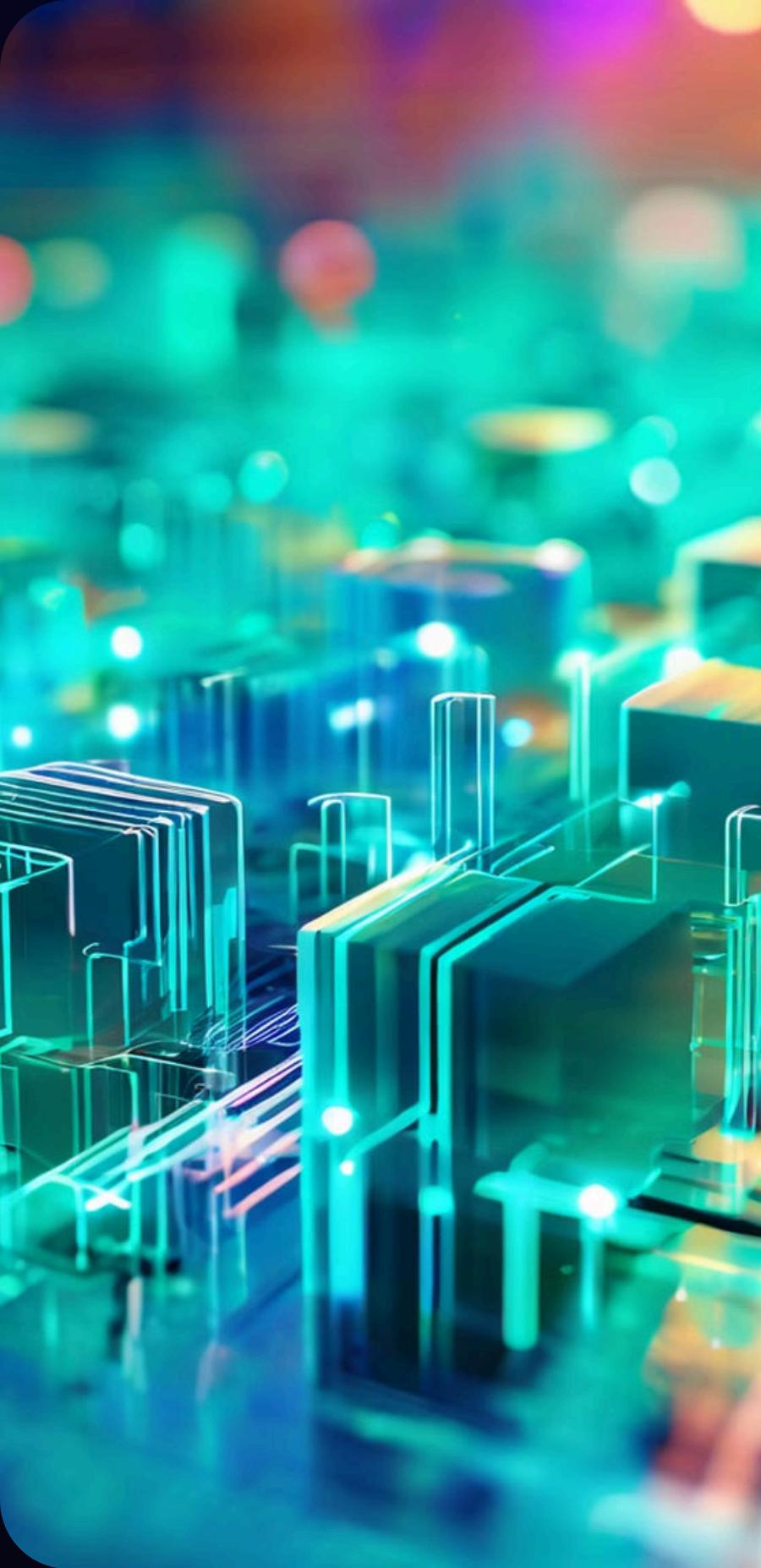
En la recursividad, una función se llama a sí misma. Cada llamada recursiva crea un nuevo marco de pila, lo que puede llevar a un desbordamiento de pila si la recursión es demasiado profunda. Es crucial implementar condiciones de base adecuadas.

3

## Optimización de Cola

La recursión de cola es una forma especial de recursividad donde la llamada recursiva es la última operación en la función. Algunos compiladores pueden optimizar esto, convirtiendo la recursión en un bucle, lo que ahorra espacio en la pila.





# Gestión de Memoria y Cadenas en C

Función	Descripción	Uso Común
malloc()	Asigna memoria del heap	Crear arreglos dinámicos
calloc()	Asigna e inicializa a cero	Matrices multidimensionales
realloc()	Redimensiona memoria asignada	Ajustar tamaño de arreglos
free()	Libera memoria asignada	Prevenir fugas de memoria
strncpy()	Copia cadenas con límite	Copiar strings de forma segura
strncat()	Concatena con límite	Unir strings sin desbordamiento



# Procesos, Subprocesos e Hilos en C



## Procesos

Los procesos son instancias de programas en ejecución con su propio espacio de memoria. En sistemas UNIX, se crean mediante `fork()`. Cada proceso tiene recursos propios y está aislado de otros procesos, lo que proporciona seguridad pero aumenta la sobrecarga de comunicación entre procesos.



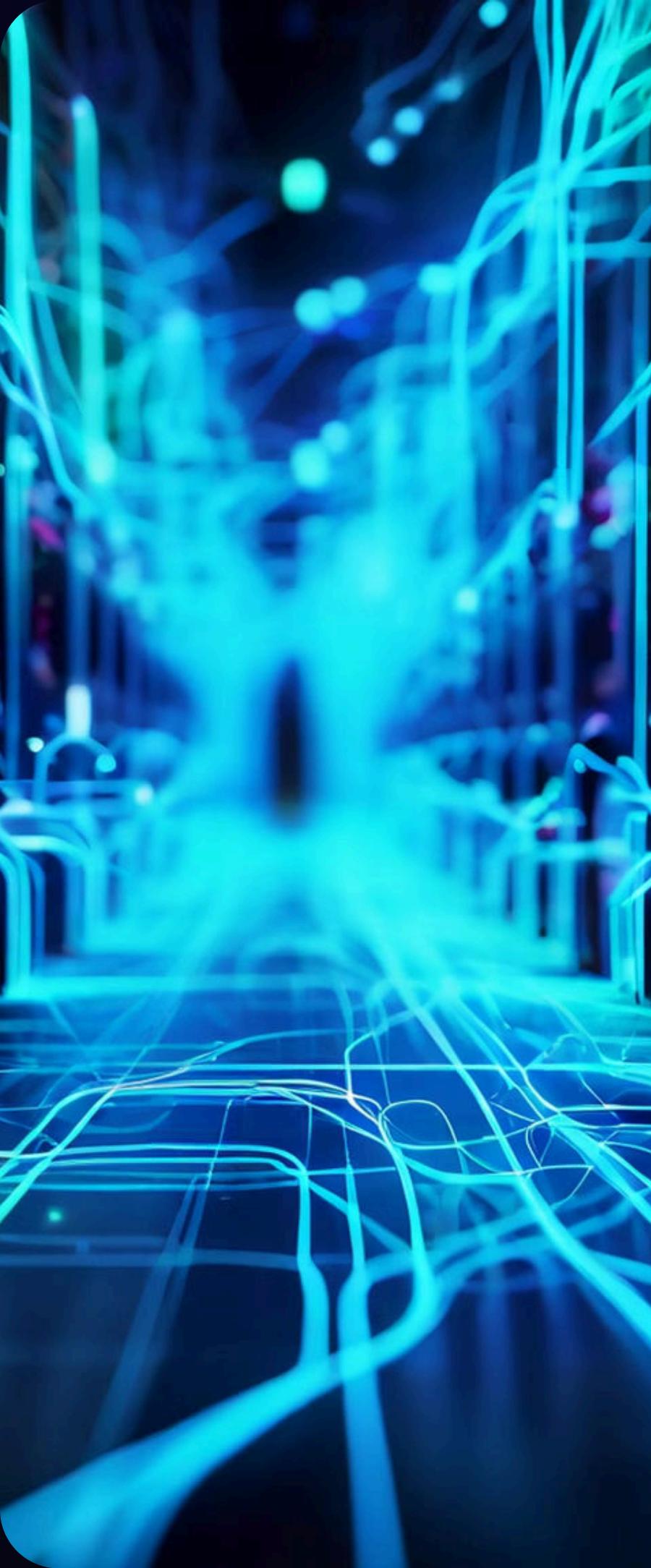
## Hilos

Los hilos son unidades de ejecución dentro de un proceso que comparten el mismo espacio de memoria. Se gestionan con la biblioteca `pthread` en C. Los hilos permiten la concurrencia dentro de un proceso, facilitando la parallelización de tareas y el uso eficiente de recursos en sistemas multiprocesador.



## Sincronización

La sincronización entre hilos es crucial para evitar condiciones de carrera. Se utilizan mecanismos como `mutex`, semáforos y variables de condición. Estos permiten coordinar el acceso a recursos compartidos y garantizar la coherencia de los datos en entornos multihilo.



# Network Sockets en C

## Creación de Sockets

Los sockets se crean con la función `socket()`. Se especifica el dominio (AF\_INET para IPv4), tipo (SOCK\_STREAM para TCP, SOCK\_DGRAM para UDP) y protocolo. Esta función devuelve un descriptor de archivo para el socket.

## Conexión y Enlace

Para servidores, `bind()` asocia el socket a una dirección. Los clientes usan `connect()` para establecer una conexión con un servidor. En servidores TCP, `listen()` prepara el socket para aceptar conexiones entrantes.

## Intercambio de Datos

`send()` y `recv()` se utilizan para enviar y recibir datos en conexiones TCP. Para UDP, se usan `sendto()` y `recvfrom()`. Estas funciones manejan la transmisión de datos a través de la red.

## Cierre de Conexiones

`close()` o `shutdown()` se usan para cerrar conexiones. `shutdown()` ofrece más control, permitiendo cerrar solo la lectura o escritura. Es importante cerrar correctamente los sockets para liberar recursos del sistema.