Análisis de la sentencia:

```
inet_ntoa(*((struct in_addr *)h->h_addr))
```

Para entenderla, hay que leerla de **adentro hacia afuera:**

Paso 1: h->h_addr

En la estructura hostent, el campo h_addr es en realidad un alias (un #define) para h_addr_list[0].

- **Qué es:** Es un apuntador de tipo char *.
- **Qué contiene:** La dirección de memoria donde empiezan los 4 bytes de la dirección IP en formato binario.
- **El problema:** Al ser un char *, el lenguaje C cree que lo que hay ahí son letras, pero nosotros sabemos que es una estructura de red.

Paso 2: (struct in_addr *)

Aquí realizamos un **casting** (una conversión explícita).

- Le estamos diciendo al compilador: *"No trates este apuntador como un simple caracter (char *), trátalo como un apuntador a una estructura in_addr"*.
- La estructura in_addr es la que Linux usa específicamente para almacenar direcciones IPv4 binarias.

Paso 3: *((struct in_addr *) ...)

El asterisco al principio es el operador de **desreferenciación**.

- **Acción:** Vamos a la dirección de memoria a la que apunta el apuntador y extraemos el valor real (la estructura completa).
- **Resultado:** Ahora ya no tenemos un apuntador (una dirección), sino el **dato binario de la IP** empaquetado en una struct in_addr.

Paso 4: inet_ntoa(...)

Finalmente, pasamos ese dato a la función inet_ntoa (Internet Network to ASCII/Address).

- **Entrada:** Recibe la struct in_addr (los 4 bytes binarios).
- **Salida:** Devuelve un puntero a una cadena de texto (ej. "192.168.1.10").