Manejo de direcciones IP en C

Tomás Fernández Pena
José Carlos Cabaleiro Domínguez

Grado en Ingeniería Informática

Universidade de Santiago de Compostela

Redes, 2º Curso GrEI

Índice

Índice

Protocolo y direcciones IP

- Protocolo de Internet (*Internet protocol*, IP): protocolo usado en Internet para encaminar paquetes de datos entre **computadores** conectados a Internet
  - Es el principal protocolo de comunicaciones en Internet
- Dirección IP (*IP address*): etiqueta numérica asignada a cada dispositivo conectado a una red que usa el protocolo IP
  - Todo dispositivo (ordenador, móvil, consola de videojuegos, electrodoméstico inteligente, etc.) que se conecte a Internet tiene que tener asignada una dirección IP
  - Esa dirección puede ser fija o ir cambiando (p. ej., cada vez que se reinicia el dispositivo)
  - Las direcciones tienen que ser únicas, aunque hay direcciones especiales que se pueden repetir (más sobre esto en la clase de teoría)

Versiones de las direcciones IP

Dos versiones:

- IPv4 (clásica): la más usada en equipos finales (ordenadores, móviles, etc.)
  - la dirección es un número entero de 32 bits sin signo
- IPv6: reemplaza a la anterior, se va introduciendo en routers y dispositivos intermedios
  - la dirección es un número entero de 128 bits sin signo

Compatibilidad hacia arriba:

- Toda dirección IPv4 tiene una IPv6 asociada (lo contrario no es cierto)

Formato de las direcciones IPv4

Una dirección IP se puede representar de dos diferentes formas:

- Formato binario: representación interna del dispositivo
  - Entero sin signo de 32 bits (4 bytes) (en C `in_addr_t` o `uint32_t`[1])
  - Ejemplo en hexadecimal:

```
in_addr_t ip = 0xC8806EC1;
```

- Formato textual: representación *legible* para humanos
  - Cadena de caracteres de longitud máxima `INET_ADDRSTRLEN`[2] (en C `char ip[INET_ADDRSTRLEN]`).
  - Las cadenas válidas como direcciones tienen un formato fijo:
    - 4 campos numéricos con valores entre `0` y `255` separados por punto
  - Ejemplo: `char ip[]="193.110.128.200";`

[1] Tipos definidos en el fichero de cabecera `inttypes.h`

[2] Macro definida en `netinet/in.h` y que vale 16

Formato de las direcciones IPv4

Formato adicional:

- Representar la IP como un array de 4 enteros sin signo de 8 bits (1 byte):

```
uint8_t ip1[4] = {193u,110u,128u,200u};
uint8_t ip2[4] = {0xC1,0x6E,0x80,0xC8};
```

Formato de las direcciones IPv4 en C

Muchas funciones C piden la IP en formato binario encapsulado en una estructura `struct in_addr`[3]

- Contiene un único campo `s_addr` de tipo `in_addr_t`

Ejemplo:

```
struct in_addr ip;
ip.s_addr=0xC8806EC1;
```

Macros predefinidas[4] para ciertas direcciones IPv4 especiales:

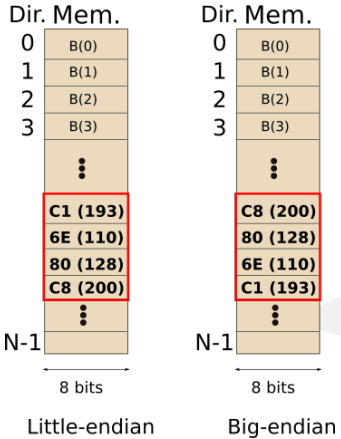
- `INADDR_LOOPBACK`: lazo de vuelta o *loopback* (127.0.0.1)
- `INADDR_ANY`: cualquier dirección válida (0.0.0.0)
- `INADDR_BROADCAST`: IP de difusión (255.255.255.255)
- `INADDR_NONE`: usado en funciones que devuelven una IP para indicar un error

[3] Declarada en `netinet/in.h`

[4] Definidas en `netinet/in.h`

Orden de host y orden de red

Dirección `in_addr_t ip = 0xC8806EC1;` en memoria



- En Internet siempre se usa big-endian (**orden de red**)
- Los dispositivos pueden usar little o big-endian (**orden de host**)

Orden de host y orden de red

En un host little-endian, los siguientes valores representan la misma IP:

- `in_addr_t ip1 = 0xC8806EC1;`
- `uint8_t ip2[4] = {0xC1,0x6E,0x80,0xC8};`
- `uint8_t ip3[4] = {193u,110u,128u,200u};`
- `char ip[]="193.110.128.200";`

Se tiene entonces que:

$$''193.110.128.200'' = 193 \times 2^0 + 110 \times 2^8 + 128 \times 2^{16} + 200 \times 2^{24} =$$

$$= 3363850161 = 0xC8806EC1$$

- En C hay funciones para realizar esa conversión (y la inversa)

Direcciones IPv6

Problema con las direcciones IPv4:

- Van de la `0x0` (0.0.0.0) a la `0xFFFFFFFF` (255.255.255.255)
- Son un total de 2^{32} valores posibles (más de 4 mil millones)
- No todas las direcciones son válidas (direcciones reservadas)
- Ya se han acabado

Soluciones

- Permitir direcciones repetidas (direcciones privadas y NAT[5])
- Migrar a IPv6 con direcciones de 128 bits (16 bytes)
  - $2^{128} = 3{,}4 \times 10^{38}$ valores posibles

[5] *Network Address Translation*: traduce direcciones privadas a públicas

Formato de las direcciones IPv6

Formato binario: representación interna del dispositivo

- Entero sin signo de 128 bits (16 bytes)
- En C se almacena encapsulado en una estructura de tipo `struct in6_addr`[6]
  - Permite acceder de diferentes formas, a través de una `union`

```
/* Formato de la estructura declarada en netinet/in.h
struct in6_addr
{ union
  {
    uint8_t s6_addr[16];
    uint16_t s6_addr16[8];
    uint32_t s6_addr32[4];
  }; } */
struct in6_addr ipv6;
ipv6.s6_addr32[0] = 0x12345678;
printf("%x\n", ipv6.s6_addr[3]); // Imprime 12
printf("%x\n", ipv6.s6_addr16[0]); // Imprime 5678
```

[6] Declarada en `netinet/in.h`

Formato de las direcciones IPv6

Formato textual: cadena de caracteres de longitud máxima `INET6_ADDRSTRLEN`[7]

- Representada mediante 8 grupos de hasta 4 dígitos hexadecimales cada uno, separados por :

```
char ip6[]="1080:0:0:0:8:800:200C:417A";
```

- Las secuencias de ceros consecutivos pueden abreviarse con ::,
  - Ejemplo 1: la dirección anterior se puede abreviar como "`1080::8:800:200C:417A`"
  - Ejemplo 2: la dirección del lazo de vuelta es "`0:0:0:0:0:0:0:1`", que se abrevia a "`::1`"
- Para facilitar el paso de IPv4 a IPv6 una dirección IPv4 puede escribirse en IPv6 como "`::FFFF:193.110.128.200`"

[7] Macro definida en el fichero de cabecera `netinet/in.h` y que vale 46

Funciones de conversión de formatos

C proporciona funciones para convertir direcciones IPv4 e IPv6 de formato textual a binario

- Dos son válidas para IPv4 e IPv6 (definidas en `arpa/inet.h`):
  - `inet_pton`: de textual a binario
  - `inet_ntop`: de binario a textual

Función `inet_pton`: textual a binario

```
int inet_pton(int af, const char *src, void *dst)
```

- Parámetros:
  - `af` entero que debe valer `AF_INET` para direcciones IPv4 o `AF_INET6` para IPv6[8]
  - `src` puntero[9] a un string con la dirección IP a convertir
  - `dst` puntero al resultado, debe apuntar a una `struct in_addr` para IPv4 o a una `struct in6_addr` para IPv6[10]
  - La IP se guarda en orden de red
- Valor devuelto: 1 en caso de éxito, 0 o -1 en caso de error

[8] Las siglas `AF` significan *address family*. Por razones históricas, a veces se usa `PF` (*protocol family*) en su lugar, pero son equivalentes.

[9] Puntero aquí significa dirección de memoria que apunta...

[10] Debe apuntar a un área reservada con el tamaño adecuado

Función `inet_pton`: textual a binario

Ejemplo de uso:

```
struct in_addr miip;
if(inet_pton(AF_INET, "193.110.128.200", (void *) &miip) != 1) {
  fprintf(stderr, "Formato de direccion incorrecto");
  exit(EXIT_FAILURE);
}
printf("%X\n", miip.s_addr); // Imprime C8806EC1
```

Función `inet_ntop`: binario a textual

```
const char *inet_ntop(int af, const void *src,
  char *dst, socklen_t size)
```

- Parámetros:
  - `af` igual que antes
  - `src` puntero[11] a una `struct in_addr` para IPv4 o a una `struct in6_addr` para IPv6
  - `dst` puntero a la cadena en la que se guardará el resultado[12]
  - `size` entero indicando el tamaño en bytes de la cadena destino[13]
- Valor devuelto: puntero a `dst`, `NULL` en caso de error

[11] Puntero aquí significa dirección de memoria que apunta...

[12] Debe apuntar a un área reservada con el tamaño adecuado

[13] `socklen_t` es un tipo de dato entero sin signo de tamaño suficiente

Función `inet_ntop`: binario a textual

Ejemplo de uso:

```
struct in_addr miip;
miip.s_addr = 0xC8806EC1;
char miiptext[INET_ADDRSTRLEN];
if(inet_ntop(AF_INET, (const void *) &miip, miiptext, INET_ADDRSTRLEN) != NULL) {
    printf("%s\n", miiptext); // Imprime 193.110.128.200
}
```

Otras funciones para IPv4

Otras funciones (solo IPv4)

- `inet_addr`: toma como entrada una dirección IPv4 en formato textual y la devuelve en formato binario sin encapsular (`in_addr_t`)
  - En caso de error devuelve `INADDR_NONE`
- `inet_aton` (`inet_ntoa`): a partir de una dirección IPv4 en formato textual obtiene el formato binario encapsulado (`struct in_addr`) (y viceversa)[14]
- `inet_network`: similar a `inet_addr`, devuelve el formato binario en *orden de host*

[14] Estas dos funciones no existen en Windows

Índice

Puertos y servicios

Puerto: número entero de que identifica de forma única un servicio (o aplicación) en un nodo final de la red

- Por ejemplo, computador ejecutando un servidor web y un servidor de e-mail (SMTP)
  - Servicio web: identificado por el puerto 80 (HTTP) o 443 (HTTPS)
  - Servicio e-mail: identificado por el puerto 25
- Un paquete de datos destinado a ese computador debe especificar el puerto (servicio) al que va dirigido

Existen números de puertos asignados a servicios concretos y otros de uso libre

- Los números van del 0 al $2^{16} - 1 = 65535$
- Tipo `uint16_t`[15]

[15] Tipo definido en el fichero de cabecera `inttypes.h`

Puertos

Macros de interés (definidas en `netinet/in.h` y `netdb.h`)

- `IPPORT_RESERVED`: puertos menores que ese valor (1024) están reservados para el superusuario y servicios estándares
- `IPPORT_USERRESERVED`: puertos mayores que este valor (5000) pueden ser usados por los usuarios
- Cuando se usa un socket sin especificar su puerto, el sistema le genera automáticamente un puerto comprendido entre `IPPORT_RESERVED` e `IPPORT_USERRESERVED`
- `NI_MAXSERV`: longitud máxima de la cadena de texto que representa a un servicio

Una lista de servicios y su puerto asociado se puede ver en el fichero `/etc/services`

Índice

Conversión del orden de los bytes

Tanto a las IPs como a los puertos les afecta el orden en el que se almacenan los bytes:

- Orden de red: debe ser siempre big-endian
- Orden de host: puede ser little o big-endian según el host

Funciones:

- `uint16_t htons(uint16_t hs)` convierte el entero de 16 bits `hs` del orden de host al orden de red.
- `uint16_t ntohs(uint16_t ns)` convierte el entero de 16 bits `ns` del orden de red al orden de host.
- `uint32_t htonl(uint32_t hl)` convierte el entero de 32 bits `hl` del orden de host al orden de red.
- `uint32_t ntohl(uint32_t nl)` convierte el entero de 32 bits `nl` del orden de red al orden de host.

Índice

Servicio de Nombres de Dominio (DNS)

Nombres de equipos

- Nombres que identifican ordenadores en Internet (p. ej., www.usc.es)
  - Más fácil de recordar que una IP
  - Son cadenas de caracteres de longitud máxima `NI_MAXHOST`[16]
- Un nombre puede tener varias direcciones IP
- Un ordenador puede tener varios nombres

[16] Macro definida en `netdb.h` y que vale 1025

Servicio de Nombres de Dominio (DNS)

El Domain Name System permite convertir nombres a direcciones IPs y viceversa

- Los navegadores consultan al DNS para obtener la IP
- El comando `dig` permiten hacer consultas al DNS
- Existen funciones en C para consultar al DNS
  - `getaddrinfo`: de nombres a direcciones
  - `getnameinfo`: de direcciones a nombres

Opciones básicas del comando `dig`:

- Obtener la IP asociada a un nombre [`aaaa` para IPv6]
- Obtener el nombre asociado a una IP [`-x`]
- Obtener los servidores DNS de un dominio [`ns`]
- Obtener los servidores de correo de un dominio [`mx`]