

# Apuntes para la implementación del cliente MINI-TELNET

Sistemas Operativos

2do Cuatrimestre - 2016

## 1. *Sockets*

Un *file descriptor*, en particular un *fd* de *socket*, tiene tipo `int`. Recordar de la clase que se puede crear un *socket* usando:

```
int socket(int domain, int type, int protocol);
```

Para trabajar con *sockets* de internet usaremos el `domain` `AF_INET`. Para trabajar con mensajes UDP (sin conexión), usaremos el `type` `SOCK_DGRAM`. Para TCP, usaremos el `type` `SOCK_STREAM`. Recordar que en `protocol` en general se utiliza un `0`.

Un socket se cierra con `close(int socket)`.

## 2. Direcciones de internet

Para representar una dirección de internet se usa la estructura presentada a continuación:

```
struct sockaddr_in {
    unsigned short sin_family; /* dominio, usamos AF_INET */
    in_port_t      sin_port;   /* número de puerto */
    struct in_addr sin_addr;    /* dirección IP */
    unsigned char  sin_zero[8]; /* padding (no se usa) */
};
```

Donde la estructura que contiene la dirección IP es la siguiente:

```
struct in_addr {
    in_addr_t s_addr; /* Esto es un número de 32 bits */
};
```

## 3. *Network byte order*

Las estructuras mencionadas arriba necesitan tener el **puerto** y la **dirección IP** almacenadas en un formato conocido como *Network byte order*<sup>1</sup>. Para ello contamos con funciones de conversión:

---

<sup>1</sup>Se trata de un estándar *big-endian*

- `uint16_t htons(uint16_t hostshort)` convierte un *uint16\_t* del *host* (máquina local) en un *uint16\_t* de la red.
- `uint32_t htonl(uint32_t hostlong)` análoga pero convierte *uint32\_t*.

## 4. Resolver direcciones IP

Para convertir una cadena de caracteres que contiene una dirección IP (por ejemplo: “127.0.0.1”) en una estructura `in_addr` usamos:

```
int inet_aton(const char *cp, struct in_addr *inp);
```

Esta función ya nos deja la dirección IP en formato *Network byte order*.

¡Ojo! Esta función devuelve 0 en caso de error (sí, es al revés que la mayoría de las funciones de sistema).

## 5. Enviar paquetes UDP

Para enviar paquetes UDP usamos la siguiente llamada al sistema:

```
ssize_t sendto(int sockfd, const void *buf, size_t len, int flags,
               const struct sockaddr *dest_addr, socklen_t addrlen);
```

Se usa el socket `s` para enviar `len` bytes de datos desde el buffer apuntado por `buf` hacia la dirección apuntada por `to`, cuya longitud es de `tolen`.

## 6. Otras funciones útiles

- `ssize_t getline(char **lineptr, size_t *n, FILE *stream);`  
Leer una línea (hasta “\n”) desde `stream`.
- `int strncmp(const char *s1, const char *s2, size_t n);`  
Comparar dos cadenas `s1` y `s2` de longitud a lo sumo `n`.
- `int system(const char *command)`  
Ejecuta el comando `command` en un shell.

## 7. Pistas extra

En el ejercicio 1, a la hora de enviar el resultado del comando de vuelta al cliente, será necesario utilizar la función `dup2(2)`.