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

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

```
struct in_addr {
    unsigned long s_addr;          // Esto es un long 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*[1]. Para ello contamos con funciones de conversión:

[1] Se trata de un estándar *big-endian*

- `unsigned short htons(unsigned short us)` convierte un *short* del *host* (máquina local) en un *short* de la red.
- `unsigned long htonl(unsigned long ul)` análoga pero convierte *longs*.

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. *Includes* recomendados

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
```

7. Otras funciones útiles

- `char* fgets(char* s, int size, FILE* stream);`
  Leer una línea (hasta "`\n`" de a lo sumo `size` 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`.

8. Pistas extra

Revisen el comando dup2 asociado a los `fd` de los streams `stdout` y `stderr`.

Investiguen el commando fflush()

Puede ver más información con `man dup2` y `man fflush`.