Sistemas Operativos

# Taller IPC: Mini-Telnet

Grupo 20


| | | |
|---|---|---|
| Daniel Grosso | 694/08 | dgrosso@gmail.com |
| Mariano De Sousa Bispo | 389/08 | marian_sabianaa@hotmail.com |


Agosto 2010

# 1. Cliente de MINI-TELNET UDP

El primer cliente programado (`client.c`) usa el protocolo `UDP` para comunicarse con el *server* dado por la cátedra (`mt_server.c`). El programa asume que recibe la dirección IP del *server* (en formato `IPv4`) en el primer argumento y la convierte en una dirección binaria (formato *Network Byte Order*) mediante la función `inet_aton`. Luego crea un *socket* con protocolo `UDP` (`SOCK_DGRAM`) y dominio `INET` usando la función `socket`. Luego, espera de forma bloqueante a que el usuario ingrese algún comando por `stdin` usando la función `fgets`. Cuando recibe el comando, lo envía al servidor a través del *socket* con la función `sendto` y espera a que el usuario ingrese un nuevo comando. Este procedimiento se repite hasta que el comando ingresado es "*chau*", comando con el cual se cierra el *socket* y termina el programa.

# 2. Cliente y servidor de MINI-TELNET TCP

## 2.1. Servidor

El servidor programado (`mt_server_tcp.c`) usa el protocolo `TCP`. Para ello, primero crea con la función `socket` el *socket* con dominio `INET` y protocolo `TCP` (`SOCK_STREAM`). Luego, asigna el puerto del *socket* con la función `bind` y pasa a escuchar conexiones entrantes de manera bloqueante en el puerto asignado (función `listen`). La primer conexión que llega al puerto es aceptada con `accept` creando un nuevo *socket* para establecer la comunicación, y espera de forma bloqueante a que el cliente envíe un comando (función `read`) a través del nuevo *socket*. Al recibir el comando, se llama a la función `popen` que ejecuta el comando especificado como proceso hijo y crea un *pipe* para redireccionar la salida estándard, devolviendo un descriptor de archivo. La salida estándard del comando es leída en un *buffer* mediante el uso de `fread`. Finalmente, envía al cliente a través del *socket* el contenido del *buffer* y repite el procedimiento hasta recibir el comando "*chau*", comando con el cual se cierran los *sockets* y termina el programa.

## 2.2. Cliente

El cliente (`client_tcp.c`) usa el protocolo `TCP` para comunicarse con el *server* (`mt_server_tcp.c`). El programa asume que recibe la dirección IP del

*server* (en formato `IPv4`) en el primer argumento y la convierte en una dirección binaria (formato *Network Byte Order*) mediante la función `inet_aton`. Luego crea un *socket* con protocolo `TCP` (`SOCK_STREAM`) y dominio `INET` con el cual intentará conectarse al servidor a través de la función `connect`. Una vez conectado, espera de forma bloqueante a que el usuario ingrese algún comando por `stdin`. Cuando recibe el comando, lo envía al servidor a través del *socket* y espera de forma bloqueante a que el servidor envíe la salida estándard del comando mediante la función `read` para luego mostrarla. Este procedimiento se repite hasta que el comando ingresado es "*chau*", comando con el cual se cierra el *socket* y termina el programa.