Práctica de comunicación entre aplicaciones por sockets

Socket designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos. El término socket es también usado como el nombre de una interfaz de programación de aplicaciones (API) para la familia de protocolos de Internet TCP/IP, provista usualmente por el sistema operativo.

Los sockets de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un socket queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

El maestro

En esta práctica un programa (master.c) se comporta como aplicación maestro (la que "manda" en la relación) o como cliente (el que solicita un servicio).

```
main(int argc, char *argv[]) {
int i;
int new_sock_id;
if(argc<3){printf("Modo de uso:  master IP port\n");exit(0);}
        printf("Opening port %d in %s..\n",atoi(argv[2]),argv[1]);

        new_sock_id=open_socket_client(atoi(argv[2]),argv[1]);
        printf("Conectado. Write words (and enter). To finish, write \"end\"\n");
        do{
                scanf("%s",mess);
                write(new_sock_id, mess, 1+strlen(mess));
        }while(strcmp(mess,"end"));
printf("Closing socket...\n");
close(new_sock_id);
}
```

Observe los siguientes detalles.

- La línea en cursiva indica que al ejecutar la aplicación debes indicar la dirección IP de la máquina a la que te tienes que conectar por sockets, así como el número de puerto. En internet, las direcciones IP identifican a cada máquina del nodo, mientras que el número de puerto es una forma de identificar a las aplicaciones dentro de la máquina.
- En la primera línea en rojo, El cliente establece la conexión a través de un socket. En la máquina (y puerto) destino, debe haber **previamente** un esclavo "a la escucha", en ese puerto, para que la llamada tenga éxito. En caso contrario, se queda bloqueado.
- En la segunda línea en negrita, el cliente envía paquetes de texto (mess) que previamente ha leído del teclado (scanf).

El esclavo o servidor.

Es la aplicación (y la máquina) que recibe las órdenes enviadas desde el maestro en forma de paquetes de texto. Observe el código:

```
main(int argc, char *argv[]) {
int i;
pthread_t mithread;    /* thread and attributes */
int sock_id, new_sock_id;
if(argc<2){printf("Modo de uso: slave port\n");exit(0);}
        printf("Opening port...\n");
        new_sock_id=open_socket(atoi(argv[1]),sock_id);
        printf("Conectado.........\n");
        pthread_create(&mithread,NULL,  unction_hilo,(void *)(new_sock_id));
        printf("Thread started \n");
        do{
                if(mess[0]==0)printf(".");
                else printf(" %s\n",mess);
                fflush(0);
                sleep(1);
        }while(strcmp(mess,"end"));
printf("Waiting for thread...\n");
pthread_join(mithread,NULL);
close(sock_id);
}
```

- La primera línea en negrita inicia el socket, que queda a la espera de un cliente. A diferencia de éste, la línea no se bloquea. El socket queda abierto.

```
main(int argc, char *argv[]) {
int i;
pthread_t mithread;    /* thread and attributes */
int sock_id, new_sock_id;
if(argc<2){printf("Modo de uso: slave port\n");exit(0);}
        printf("Opening port...\n");
        new_sock_id=open_socket(atoi(argv[1]),sock_id);
        printf("Conectado.........\n");
        pthread_create(&mithread,NULL,  unction_hilo,(void *)(new_sock_id));
        printf("Thread started \n");
        do{
                if(mess[0]==0)printf(".");
                else printf(" %s\n",mess);
                fflush(0);
                sleep(1);
        }while(strcmp(mess,"end"));
printf("Waiting for thread...\n");
pthread_join(mithread,NULL);
close(sock_id);
}
```

- En este ejemplo, y una vez abierto el socket de comunicación, el esclavo genera un hilo (línea en rojo) que es el que realiza la escucha de mensajes.

- Además, el hilo principal (lazo do-while) realiza un trabajo, utilizando los mensajes que puedan llegar del cliente. Si no hay mensajes (`mess[0]==0`), imprime un punto. Si hay mensajes, lo imprime. Si el mensaje es la palabra *end*, finaliza la comunicación.
- A continuación, observe la rutina con el pthread:

```
void *funcion_hilo(void* new_sock_id) {
int n=0;
        do{
                sleep(1);
                bzero(mess,100);
                n = read(new_sock_id,mess,100);
                printf("\nEl thread ha recibido  %d bytes:",n);
                if (n < 0) perror("ERROR leyendo del socket");
        }while(strcmp(mess,"end"));
printf("\nThread terminando...\n");
pthread_exit(NULL);
return 0;
}
```

Para probar, compile y ejecute el esclavo, con un puerto al azar (como 1500). Observe que puede haber puertos no disponibles, que provocarían un error:

- gcc -o slave slave.c –lpthread
- ./slave 1500 &

Observe el símbolo & que deja la aplicación ejecutándose en segundo plano. A continuación, compile (desde la misma máquina) y ejecute el maestro:

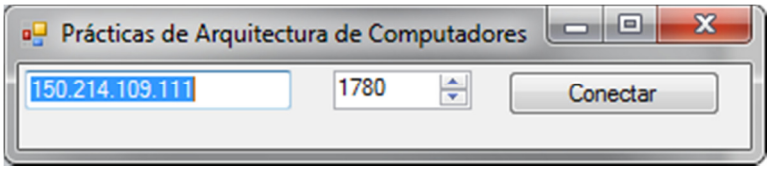
- gcc -o master master.c
- ./master 127.0.0.1 1500

127.0.0.1 es la dirección IP universal para indicar “la propia máquina”. También puede utilizar la IP de la máquina (150.214.109.111) o sus nombres de dominio: (bambu, bambu.ac.uma.es).

Envíe palabras desde el maestro, y envía la palabra *end* para finalizar maestro y esclavo.

Maestro en una máquina remota Windows

La aplicación SocketClient.exe, en la figura, realiza la función del cliente en una máquina Windows



Se adjunta el proyecto Visual Studio, en la que se puede leer el código que se ejecuta al pulsar el botón conectar:

```
try
{
   Cliente= new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
   IPEndPoint remoteEP= new IPEndPoint(
               IPAddress.Parse(textBox2.Text),
               (int)numericUpDown1.Value);
   Cliente.Connect(remoteEP);
}
catch {}
finally
{
       conectado = Cliente.Connected;
}
```

Se observa la creación del Socket, y la conexión a una aplicación remota, definida por la clase remoteEP, que a su vez se declara indicando la dirección IP y el puerto (previamente establecidos en dos controles de tipo textbox y numericupdown).

Ponga el marcha el esclavo en bambú, así como el cliente en Windows.