



Instituto Politécnico Nacional

Escuela Superior de Cómputo



## Desarrollo de Sistemas Distribuidos

“Practica: Clase Sockets UDP en C++”

**Profesor:** Coronilla Contreras Ukranio

**Grupo:** 4CM4

**Equipo 6:**

- Grimaldo Peralta Marco Antonio
- Huerta Cortés Alan Antonio
- Macías Castillo Josue
- Vilches Segundo Galilea Yanely

## Código fuente.

### PaqueteDatagrama.h

```
• #ifndef PAQUETEDATAGRAMA_H_
• #define PAQUETEDATAGRAMA_H_
•
• class PaqueteDatagrama
• {
•     public:
•         PaqueteDatagrama(char *, unsigned int, char *, int );
•         PaqueteDatagrama(unsigned int );
•         ~PaqueteDatagrama();
•         char *obtieneDireccion();
•         unsigned int obtieneLongitud();
•         int obtienePuerto();
•         char *obtieneDatos();
•         void inicializaPuerto(int);
•         void inicializaIp(char *);
•         void inicializaDatos(char *);
•     private:
•         char *datos; //Almacena los datos
•         char ip[16]; //Almacena la IP
•         unsigned int longitud; //Almacena la longitud de la cadena de datos
•         int puerto; //Almacena el puerto
• };
•
• #endif
```

### PaqueteDatagrama.cpp

```
• #include <string.h>
• #include <cstdlib>
• #include "PaqueteDatagrama.h"
• using namespace std;
•
• PaqueteDatagrama::PaqueteDatagrama(char* data, unsigned int datatam, char*
dir, int port) {
•     datos = new char[datatam];
•     longitud = datatam;
•     memcpy(datos, data, longitud);
•     memcpy(ip, dir, sizeof(ip));
•     puerto = port;
• }
•
• PaqueteDatagrama::PaqueteDatagrama(unsigned int tam) {
•     datos = new char[tam];
•     longitud = tam;
• }
•
• PaqueteDatagrama::~~PaqueteDatagrama() {
•     delete[] datos;
```

```

•     longitud = 0;
•     puerto = 0;
• }
•
• char* PaqueteDatagrama::obtieneDireccion() {
•     return ip;
• }
•
• unsigned int PaqueteDatagrama::obtieneLongitud() {
•     return longitud;
• }
•
• int PaqueteDatagrama::obtienePuerto() {
•     return puerto;
• }
•
• char* PaqueteDatagrama::obtieneDatos() {
•     return datos;
• }
•
• void PaqueteDatagrama::inicializaPuerto(int port) {
•     puerto = port;
• }
•
• void PaqueteDatagrama::inicializaIp(char* dir) {
•     memcpy(ip, dir, 16);
• }
•
• void PaqueteDatagrama::inicializaDatos(char* data) {
•     memcpy(datos, data, longitud);
• }

```

## SocketDatagrama.h

```

• #ifndef SOCKETDATAGRAMA_H_
• #define SOCKETDATAGRAMA_H_
•
• #include "PaqueteDatagrama.h"
• #include <netinet/in.h>
•
• class SocketDatagrama {
•     public:
•         SocketDatagrama(int);
•         ~SocketDatagrama();
•         //Recibe un paquete tipo datagrama proveniente de este socket
•         int recibe(PaqueteDatagrama & p);
•         //Envía un paquete tipo datagrama desde este socket
•         int envia(PaqueteDatagrama & p);
•     private:
•         struct sockaddr_in direccionLocal;
•         struct sockaddr_in direccionForanea;
•         int s; //ID socket
• };

```

```
•  
• #endif
```

## SocketDatagrama.cpp

```
• #include "PaqueteDatagrama.h"  
• #include "SocketDatagrama.h"  
• #include <iostream>  
• #include <cstring>  
• #include <stdio.h>  
• #include <stdlib.h>  
• #include <netinet/in.h>  
• #include <arpa/inet.h>  
• #include <unistd.h>  
• using namespace std;  
•  
• SocketDatagrama::SocketDatagrama(int pto) {  
•     bzero((char *)&direccionForanea, sizeof(direccionForanea));  
•     bzero((char *)&direccionLocal, sizeof(direccionLocal));  
•  
•     if ((s = socket(AF_INET, SOCK_DGRAM, 0)) < 0) {  
•         perror("socket creation failed");  
•         exit(EXIT_FAILURE);  
•     }  
•  
•     direccionLocal.sin_family = AF_INET;  
•     direccionLocal.sin_addr.s_addr = INADDR_ANY;  
•  
•     if(pto == 0)  
•         direccionLocal.sin_port = htons(pto);  
•     else  
•         direccionLocal.sin_port = pto;  
•  
•     if (bind(s, (struct sockaddr *)&direccionLocal, sizeof(direccionLocal))  
• < 0 ) {  
•         perror("bind failed");  
•         exit(EXIT_FAILURE);  
•     }  
• }  
•  
• SocketDatagrama::~~SocketDatagrama() {  
•     close(s);  
• }  
•  
• int SocketDatagrama::recibe(PaqueteDatagrama &p) {  
•     unsigned int len = sizeof(direccionForanea);  
•     int rec = recvfrom(s, (char *)p.obtieneDatos(), p.obtieneLongitud() *  
• sizeof(char), 0, (struct sockaddr *) &direccionForanea, &len);  
•     unsigned char ip[4];  
•     memcpy(ip, &direccionForanea.sin_addr.s_addr, 4);  
•  
•     string ip1 = to_string(ip[0]);  
•     string ip2 = to_string(ip[1]);
```

```

•     string ip3 = to_string(ip[2]);
•     string ip4 = to_string(ip[3]);
•
•     ip1.append(".");
•     ip1.append(ip2);
•     ip1.append(".");
•     ip1.append(ip3);
•     ip1.append(".");
•     ip1.append(ip4);
•
•     char dirIp[16];
•     strcpy(dirIp, ip1.c_str());
•     p.inicializaIp(dirIp);
•     p.inicializaPuerto(direccionForanea.sin_port);
•     return rec;
• }
•
• int SocketDatagrama::envia(PaqueteDatagrama &p) {
•     direccionForanea.sin_family = AF_INET;
•     direccionForanea.sin_addr.s_addr = inet_addr(p.obtieneDireccion());
•     direccionForanea.sin_port = p.obtienePuerto();
•     return sendto(s, (char *)p.obtieneDatos(), p.obtieneLongitud() *
sizeof(char), 0, (struct sockaddr *) &direccionForanea,
sizeof(direccionForanea));
• }

```

## server.cpp

```

• #include "PaqueteDatagrama.h"
• #include "SocketDatagrama.h"
• #include <iostream>
• using namespace std;
•
• int main() {
•     SocketDatagrama server = SocketDatagrama(7200);
•     cout << "Server iniciado" << endl;
•
•     while(1) {
•         PaqueteDatagrama paquete = PaqueteDatagrama(65507);
•
•         if(server.recibe(paquete)) {
•             cout << "\nMensaje recibido de:" << endl;
•             cout << "IP: " << paquete.obtieneDireccion() << endl;
•             cout << "Puerto: " << paquete.obtienePuerto() << endl;
•             cout << "Datos: " << paquete.obtieneDatos() << endl;
•             cout << "Longitud: " << paquete.obtieneLongitud() << endl;
•         }
•     }
• }

```

## client.cpp

```

• #include "PaqueteDatagrama.h"
• #include "SocketDatagrama.h"
• #include <string.h>
• #include <iostream>
• using namespace std;
•
• int main(int argc, char* argv[]){
•     if(argc != 2) {
•         printf("Forma de uso: %s ip_servidor\n", argv[0]);
•         exit(0);
•     }
•
•     char *msgChar;
•     string msg;
•     cout << ">";
•     cin >> msg;
•
•     strcpy(msgChar,msg.c_str());
•
•     memcpy(msgChar, msg.c_str(), strlen(msg.c_str())+1);
•
•     SocketDatagrama client = SocketDatagrama(0);
•     PaqueteDatagrama dat = PaqueteDatagrama(msgChar, strlen(msg.c_str()) ,
• argv[1], 7200);
•     if(client.envia(dat)){
•         cout << "Mensaje enviado a:" << endl;
•         cout << "IP: " << dat.obtieneDireccion() << endl;
•         cout << "Puerto: " << dat.obtienePuerto() << endl;
•         cout << "Longitud: " << dat.obtieneLongitud() << endl;
•     }
•
•     PaqueteDatagrama res = PaqueteDatagrama(4);
•     if(client.recibe(res)){
•         cout << "\nMensaje recibido de:" << endl;
•         cout << "IP: " << res.obtieneDireccion() << endl;
•         cout << "Puerto: " << res.obtienePuerto() << endl;
•         int res2;
•         memcpy(&res2, res.obtieneDatos(), 4);
•         printf("Respuesta: %d\n", res2);
•         cout << "Longitud: " << res.obtieneLongitud() << endl;
•     }
•     return 0;
• }

```

## Makefile

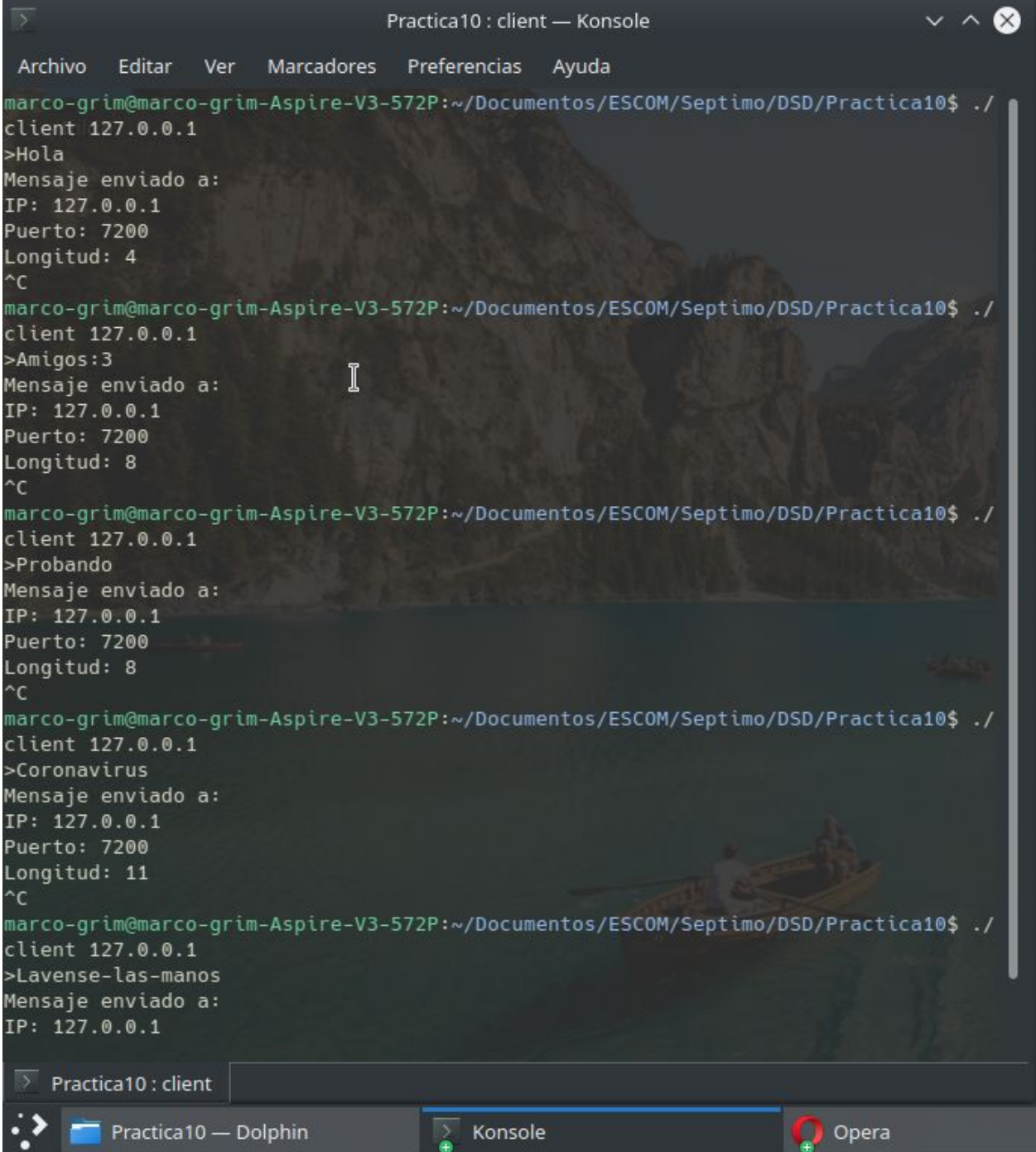
```

• run: client.cpp server.cpp SocketDatagrama.o
•     g++ server.cpp SocketDatagrama.o PaqueteDatagrama.o -o server
•     g++ client.cpp SocketDatagrama.o PaqueteDatagrama.o -o client
•
• SocketDatagrama.o: SocketDatagrama.cpp PaqueteDatagrama.o SocketDatagrama.h
•     g++ SocketDatagrama.cpp -c
•
• PaqueteDatagrama.o: PaqueteDatagrama.cpp PaqueteDatagrama.h
•     g++ PaqueteDatagrama.cpp -c

```

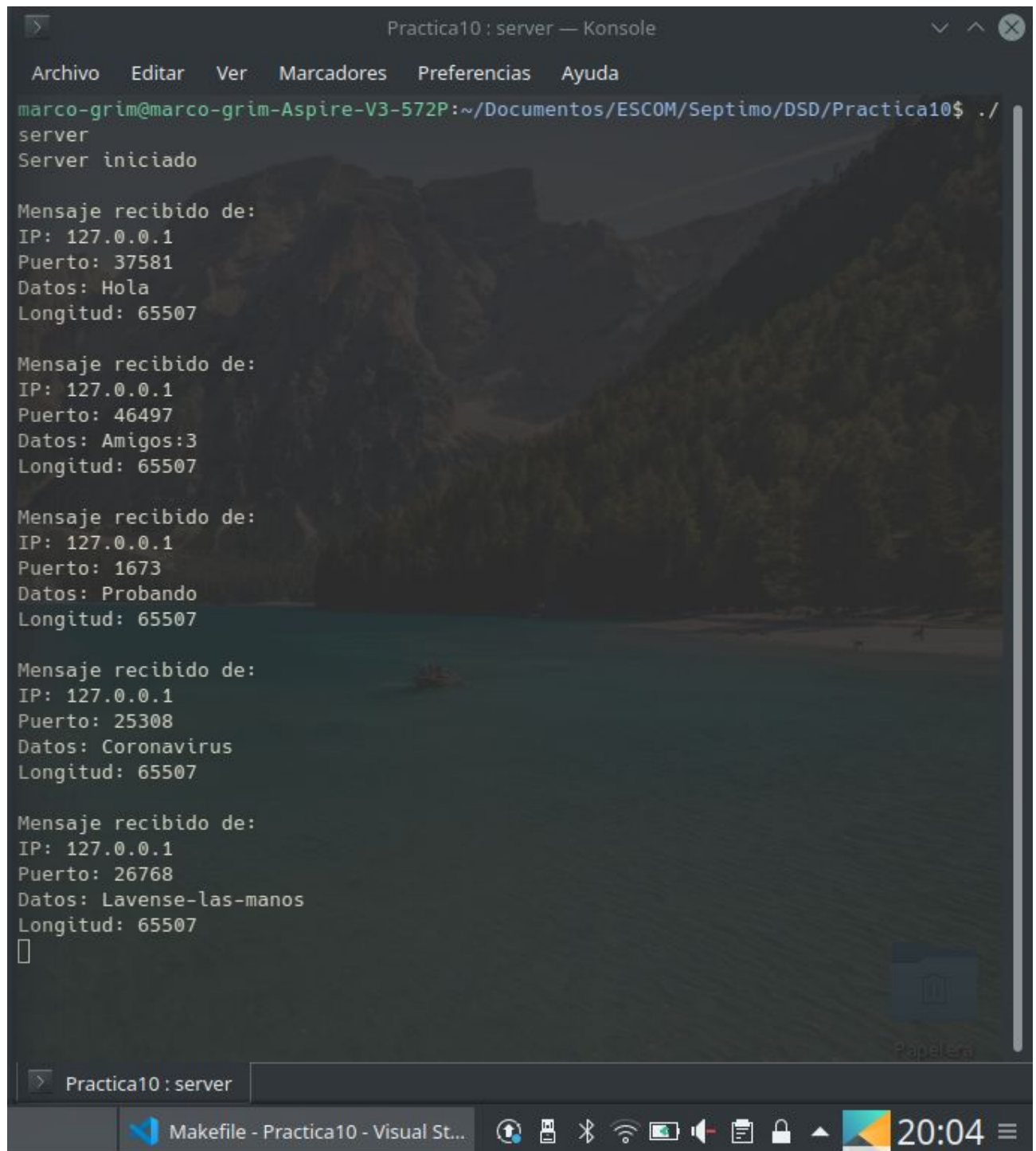
## Captura de pantalla de los resultados

### Cliente



```
Practica10 : client — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
marco-grim@marco-grim-Aspire-V3-572P:~/Documentos/ESCOM/Septimo/DSD/Practica10$ ./
client 127.0.0.1
>Hola
Mensaje enviado a:
IP: 127.0.0.1
Puerto: 7200
Longitud: 4
^C
marco-grim@marco-grim-Aspire-V3-572P:~/Documentos/ESCOM/Septimo/DSD/Practica10$ ./
client 127.0.0.1
>Amigos:3
Mensaje enviado a:
IP: 127.0.0.1
Puerto: 7200
Longitud: 8
^C
marco-grim@marco-grim-Aspire-V3-572P:~/Documentos/ESCOM/Septimo/DSD/Practica10$ ./
client 127.0.0.1
>Probando
Mensaje enviado a:
IP: 127.0.0.1
Puerto: 7200
Longitud: 8
^C
marco-grim@marco-grim-Aspire-V3-572P:~/Documentos/ESCOM/Septimo/DSD/Practica10$ ./
client 127.0.0.1
>Coronavirus
Mensaje enviado a:
IP: 127.0.0.1
Puerto: 7200
Longitud: 11
^C
marco-grim@marco-grim-Aspire-V3-572P:~/Documentos/ESCOM/Septimo/DSD/Practica10$ ./
client 127.0.0.1
>Lavense-las-manos
Mensaje enviado a:
IP: 127.0.0.1
```

## Servidor



```
Practica10 : server — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
marco-grim@marco-grim-Aspire-V3-572P:~/Documentos/ESCOM/Septimo/DSD/Practica10$ ./
server
Server iniciado

Mensaje recibido de:
IP: 127.0.0.1
Puerto: 37581
Datos: Hola
Longitud: 65507

Mensaje recibido de:
IP: 127.0.0.1
Puerto: 46497
Datos: Amigos:3
Longitud: 65507

Mensaje recibido de:
IP: 127.0.0.1
Puerto: 1673
Datos: Probando
Longitud: 65507

Mensaje recibido de:
IP: 127.0.0.1
Puerto: 25308
Datos: Coronavirus
Longitud: 65507

Mensaje recibido de:
IP: 127.0.0.1
Puerto: 26768
Datos: Lavense-las-manos
Longitud: 65507
█

Practica10 : server
Makefile - Practica10 - Visual St... 20:04
```