Ejercicio Sockets Suma Resta

Siguiendo con nuestra propuesta de construir embriones de servicios, como hemos hecho en la práctica programando un embrión de Telnet, un embrión de cliente de un servidor Apache, hoy intentarán construir un embrión de un Servidor de nombres de dominios o DNS

Descripción

Nuestro Servidor de DNS, tendrá que resolver el nombre del servidor de suma o el nombre servidor de resta según corresponda, para ello el servidor tiene asociado al nombre del servidor de suma su dirección IP, lo mismo que para el nombre del servidor de resta.

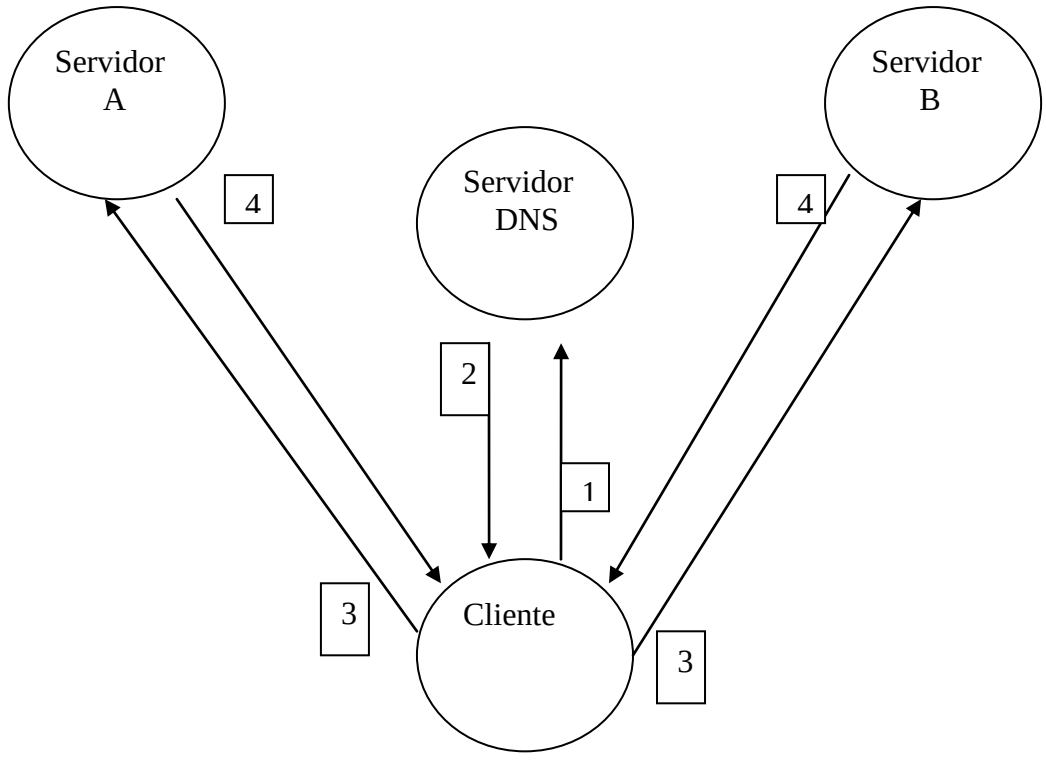
Prueba

Para probar el funcionamiento del servidor de nombres, hay que programar dos servidores uno que sume dos números y otro que reste dos números y programar un cliente que se conecte con el servidor de nombres para pedir el IP del servidor que suma o el servidor que resta.

El cliente se ejecuta desde la línea de comando de la siguiente forma:

$ ./cliente 3 + 4 o $ ./cliente 8 - 7

Diagrama del Planteo



Descripción del Planteo

Primero el cliente se conecta con el servidor DNS para obtener el IP del servidor de suma o resta según la operación que necesite el cliente, luego, una vez obtenida la dirección IP del servidor de suma o del servidor de resta, se conecta con el servidor de suma o el servidor de resta para que realizar la operación.

Servidor DNS: Recibe del cliente el nombre de dominio del servidor de suma o el nombre de dominio del servidor de resta, el servidor le envía al cliente, el IP del nombre de dominio correspondiente.

Cliente: Para conectarse con el servidor de suma o resta necesita obtener la dirección IP del mismo a través del servidor de nombres. Si quiere sumar deberá llamar al servidor de suma y si quiere restar deberá hacerlo con el servidor de resta.

Servidor de suma: Recibe dos números y devuelve el resultado de la suma, debe ser concurrente con fork o threads.

Servidor de resta: Recibe dos números y devuelve el resultado de la resta, debe ser concurrente con fork o threads.

Implementación

Se puede implementar en cualquier sistema operativo y en cualquier lenguaje la única restricción es usar sockets de tipo AF_INET. Los procesos deberían ser ejecutados en un entorno distribuido.

Solución

Servidor de nombres de dominio

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

#define IPserDNS "127.0.0.1"
#define PUERTOserDNS 6666

#define IPsuma "127.0.0.1"
#define Puertosuma "7777"

#define IPresta "127.0.0.1"
#define Puertoresta "8888"
```

```
int main(int argc, char *argv[])
{
  struct sockaddr_in s_sock,c_sock;
  int idsocks,idsockc;
  int lensock = sizeof(struct sockaddr_in);
  idsocks = socket(AF_INET, SOCK_STREAM, 0);
  printf("idsocks %d\n",idsocks);
 s_sock.sin_family      = AF_INET;
 s_sock.sin_port        = htons(PUERTOserDNS);
 s_sock.sin_addr.s_addr = inet_addr(IPserDNS);
 memset(s_sock.sin_zero,0,8);
 printf("bind %d\n", bind(idsocks,(struct sockaddr *) &s_sock,lensock));
 printf("listen %d\n",listen(idsocks,5));
  while(1)
    {
      printf("esperando conexion\n");
      idsockc = accept(idsocks,(struct sockaddr *)      &c_sock,&lensock);
      if(idsockc != -1)
        {
         printf("conexion aceptada desde el cliente\n");
         char buf[30];
         char ip[16];
         char puerto[5];
         int nb;
         nb = read(idsockc,buf,30);
         buf[nb]='\0';
         printf(".......recibido del cliente %d : %s\n",idsockc,buf);
         if (buf[0]=='+')
           {
            strcpy(ip,IPsuma);
            strcpy(puerto,Puertosuma);
           }
         else
         if (buf[0]=='-')
           {
            strcpy(ip,IPresta);
            strcpy(puerto,Puertoresta);
           }
         else
           strcpy(ip,"error");
         write(idsockc,ip,16);
         write(idsockc,puerto,5);
         close(idsockc);
        }
     else
        {
          printf("conexion rechazada %d \n",idsockc);
        }
    }
return 0 ; }
```

Servidor de suma

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

#define IPserSuma "127.0.0.1"
#define PUERTOserSuma 7777

int main(int argc, char *argv[])
{
 struct sockaddr_in s_sock,c_sock;
 int idsocks,idsockc;
 int lensock = sizeof(struct sockaddr_in);
 idsocks = socket(AF_INET, SOCK_STREAM, 0);
 printf("idsocks %d\n",idsocks);
s_sock.sin_family      = AF_INET;
s_sock.sin_port        = htons(PUERTOserSuma);
s_sock.sin_addr.s_addr = inet_addr(IPserSuma);
memset(s_sock.sin_zero,0,8);
printf("bind %d\n", bind(idsocks,(struct sockaddr *) &s_sock,lensock));
printf("listen %d\n",listen(idsocks,5));

 while(1)
  {
    printf("Servidor de suma esperando conexion\n");
    idsockc = accept(idsocks,(struct sockaddr *)&c_sock,&lensock);
    if(idsockc != -1)
      {
        printf("Servidor de suma conexion aceptada desde el cliente\n");
        int numero1 = 0 ;
        int numero2 = 0 ;
        char num1[16];
        char num2[16];
        char suma[16];
        int nb1,nb2;
        sleep(1);
        nb1 = read(idsockc,num1,16);
        sleep(1);
        nb2 = read(idsockc,num2,16);
        num1[nb1]='\0';
        num2[nb2]='\0';
        printf("servidor de suma recibio del cliente %d : %s %s \n",idsockc,num1,num2);
        numero1 = atoi(num1);
```

```
          numero2 = atoi(num2);
          sprintf(suma,"%d",numero1+numero2);
          printf("servidor de suma resultado  : %s \n",suma);
          write(idsockc,suma,strlen(suma));
          close(idsockc);
        }
      else
        {
          printf("conexion rechazada %d \n",idsockc);
        }
    }
  return 0 ;
}
```

Servidor de resta

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

#define IPserResta "127.0.0.1"
#define PUERTOserResta 8888

int main(int argc, char *argv[])
{
  struct sockaddr_in s_sock,c_sock;
  int idsocks,idsockc;
  int lensock = sizeof(struct sockaddr_in);
  idsocks = socket(AF_INET, SOCK_STREAM, 0);
  printf("idsocks %d\n",idsocks);
 s_sock.sin_family      = AF_INET;
 s_sock.sin_port        = htons(PUERTOserResta);
 s_sock.sin_addr.s_addr = inet_addr(IPserResta);
 memset(s_sock.sin_zero,0,8);
 printf("bind %d\n", bind(idsocks,(struct sockaddr *) &s_sock,lensock));
 printf("listen %d\n",listen(idsocks,5));
  while(1)
    {
      printf("Servidor de resta esperando conexion\n");
      idsockc = accept(idsocks,(struct sockaddr *)&c_sock,&lensock);
      if(idsockc != -1)
        {
          printf("Servidor de resta conexion aceptada desde el cliente\n");
```

```
          int numero1 = 0 ;
          int numero2 = 0 ;
          char num1[16];
          char num2[16];
          char resta[16];
          int nb1,nb2;
          sleep(1);
          nb1 = read(idsockc,num1,16);
          sleep(1);
          nb2 = read(idsockc,num2,16);
          num1[nb1]='\0';
          num2[nb2]='\0';
          printf("servidor de resta recibio del cliente %d : %s %s \n",idsockc,num1,num2);
          numero1 = atoi(num1);
          numero2 = atoi(num2);
          sprintf(resta,"%d",numero1-numero2);
          printf("servidor de resta resultado  : %s \n",resta);
          write(idsockc,resta,strlen(resta));
          close(idsockc);
        }
      else
        {
          printf("conexion rechazada %d \n",idsockc);
        }
    }
  return 0 ;
}
```

Cliente

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

void servidorCalculo(char [], char [],char [], char []);

#define IPserDNS "127.0.0.1"
#define PUERTOserDNS 6666

int main (int argc, char *argv[])
 {
   int sd;
   struct sockaddr_in pin;
   pin.sin_family = AF_INET;
```

```
   pin.sin_addr.s_addr = inet_addr(IPserDNS);
   pin.sin_port = htons(PUERTOserDNS);
   bzero(&pin.sin_zero, sizeof(pin.sin_zero));
   if ((sd=socket(AF_INET,SOCK_STREAM,0)) == -1)
    {
              printf("Error al abrir el socket\n");
              exit(1);
    }
    if(connect(sd, (void *)&pin,sizeof(pin)) == -1)
        {
          printf("Error al conectar el socket\n");
          exit(1);
        }
     int nb ;
     char ip[16];
     char puerto[5];
     write(sd,argv[2],2);
     sleep(1);
     nb = read(sd,ip,16);
     ip[nb];
     nb = read(sd,puerto,5);
     puerto[nb];
     close(sd);

     if (argv[2][0]=='+')
         printf("ip de suma : %s puerto %s\n",ip,puerto);

     if (argv[2][0]=='-')
         printf("ip de resta : %s puerto %s\n",ip,puerto);

     servidorCalculo(ip,puerto,argv[1],argv[3]);
     return EXIT_SUCCESS;
}


void servidorCalculo(char ip[], char puerto[],char num1[], char num2[])
{
   int sd;
   struct sockaddr_in pin;
   pin.sin_family = AF_INET;
   pin.sin_addr.s_addr = inet_addr(ip);
   pin.sin_port = htons(atoi(puerto));
   bzero(&pin.sin_zero, sizeof(pin.sin_zero));
   if ((sd=socket(AF_INET,SOCK_STREAM,0)) == -1)
    {
              printf("Error al abrir el socket\n");
              exit(1);
    }
    if(connect(sd, (void *)&pin,sizeof(pin)) == -1)
        {
```

```
            printf("Error al conectar el socket\n");
            exit(1);
          }

      char resultado[16];
       int nb ;
       printf("cliente : %s %s \n",num1,num2);
      write(sd,num1,strlen(num1));
       sleep(1);
       write(sd,num2,strlen(num2));
       sleep(1);
       nb = read(sd,resultado,strlen(resultado));
       resultado[nb]='\0';
      printf("cliente recibio resultado : %s \n",resultado);
      close(sd);
}
```