Ejercicio RPC

Servidor de Direcciones IP

Siguiendo con nuestra propuesta de construir embriones de servicios, como hemos hecho en la práctica programando un embrión de servidor de procesos y servidor de archivos, hoy intentarán construir un embrión de un Servidor de ip.

Descripción

Nuestro Servidor de ip, tendrá que resolver el nombre del host, y devolver la o las ip al cliente.

Prueba

Para probar el funcionamiento del servidor de ip, hay que programar utilizando RPC un servidor que reciba un nombre de host desde el cliente y que el servidor le devuelva al cliente la o las ip del host.

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

$ ./servidor &

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

$ ./cliente 127.0.0.1 www.google.com

Ayuda

Esta función permite obtener y muestra las direcciones IP de un nombre de Host

```
#include <netdb.h>

void funcionip(char * hostname)
{
   struct in_addr **addr_list;
   struct hostent *he = gethostbyname(hostname);

   if (he)
     {
       int i;
       puts(he->h_name);
       printf("Nombre del Host : %s\n", he->h_name);
       printf(" Direcciones IP  ");
       addr_list = (struct in_addr **)he->h_addr_list;
       for(i = 0; addr_list[i] != NULL; i++)
          {
           printf("%s \n", inet_ntoa(*addr_list[i]));
          }
       printf("\n");
     }
   else
       printf("error gethostbyname\n");
}
```

Implementación

Se debe implementar utilizando RPC

Solución

ips.x

```
struct envia
{
   char dominio[100] ;
};

struct retorno
{
   char retips[100] ;
};

program IPS_PROGRAMA{
```

```
   version VERSION_IPS_PROGRAMA{
      struct retorno ips (struct envia) = 1;
  }  =  1;
} = 0x20000001;
```

ips_client.c

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "ips.h"


void
ips_programa_1(char *host, char * dominio)
{
        CLIENT *clnt;
        struct retorno  *result_1;
        struct envia  ips_1_arg;

#ifndefDEBUG
        clnt = clnt_create (host, IPS_PROGRAMA, VERSION_IPS_PROGRAMA,
"udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
#endif  /* DEBUG */

      strcpy(ips_1_arg.dominio,dominio);

        result_1 = ips_1(&ips_1_arg, clnt);
        if (result_1 == (struct retorno *) NULL) {
                clnt_perror (clnt, "call failed");
        }

      printf("recibido del servidor %s \n", result_1->retips);

#ifndefDEBUG
        clnt_destroy (clnt);
#endif   /* DEBUG */
}
```

```
int
main (int argc, char *argv[])
{
        char *host;

        if (argc < 2) {
                printf ("usage: %s server_host dominio\n", argv[0]);
                exit (1);
        }
        host = argv[1];
        ips_programa_1 (host,argv[2]);
    exit (0);
}
```

ips_server.c

```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "ips.h"
#include <netdb.h>
#include <string.h>

void funcionip(char * ,char []);

struct retorno * ips_1_svc(struct envia *argp, struct svc_req *rqstp)
{
        static struct retorno  result;

    memset(result.retips,0,100);
        funcionip(argp->dominio,result.retips);

        return &result;
}



void funcionip(char * dominio, char TodasLasIp[])
{
   struct in_addr **addr_list;
   struct hostent *he = gethostbyname(dominio);

   if (he)
     {
       int i;
       puts(he->h_name);
```

```
      printf("Servidor Dice Nombre del Host : %s\n", he->h_name);
      printf("Servidor Dice Direcciones IP  ");
      addr_list = (struct in_addr **)he->h_addr_list;
      for(i = 0; addr_list[i] != NULL; i++)
         {
          printf("%s \n", inet_ntoa(*addr_list[i]));
          strcat(TodasLasIp,(char*) inet_ntoa(*addr_list[i]));
          strcat(TodasLasIp,"\n");
         }
      printf("\n");
    }
   else
    {
      printf("Servidor dice error gethostbyname\n");
      strcat(TodasLasIp,"error gethostbyname\n");
    }
}
```