Ejemplo de codificación bajo esquema RPC

Enfoque tradicional de llamadas locales en un programa

- #include "rand.h"
-
- void **initialize_random** (long int semilla)
- {
  - srand48(semilla);
- }
-
- double **get_netx_random** (void)
- {
  - return (drand48(void));
- }

Enfoque tradicional de llamadas locales en un programa

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include "rand.h"

int main (int argc, char *argv[])
{
   int iters;    /* Cantidad de iteraciones */
   int i;
   long semilla;      /* Semilla a utilizar */

   If (argc != 3)
   {
      fprintf (stderr, "Uso: %s semilla iteraciones\n", argv[0]);
      exit (1);
   }

   semilla=(long) atoi (argv[1]);
   iters=atoi(argv[2]);
   initialize_random(semilla);
   for (i=0; i<iters; i++)
      printf("%d: %f\n", i, get_next_random());
   exit (0);
}
```

Llamadas remotas a usar

```
#include <stdlib.h>

void initialize_random (long semilla);
double get_next_random (void);
```

El archivo de especificación para RPCGEN

```
/*   rand.x   */

program RAND_PROG {
    version RAND_VERS {
    void INITIALIZE_RANDOM(long)=1;
    double GET_NEXT_RANDOM(void)=2;
    }=1;
}= 0x31111111;
```

Orden para activar RPCGEN

```
$ rpcgen -C -a rand.x
```

- Opción “C” indica ANSI C
- Opción “a” indica que deben producir plantillas de programas para el cliente y el servidor

- “rand.h” es modificado y se generan los talones

¿Qué produce RPCGEN?

- El archivo "**rand_client.c**" contiene el código generado por "rpcgen" para la aplicación del cliente y *deberá ser alterado* por el programador.

- El archivo "**rand_clnt.c**" contiene el código del talón vinculado con el cliente, que "rpcgen" generó y *no* debe ser alterado.

- El archivo "**rand_server.c**" registra la plantilla con el código generado por "rpcgen" para la aplicación del servidor y *deberá ser alterado*.

- El archivo "**rand_svc.c**" ha sido producto de "rpcgen" y almacena el código del talón del servidor. *No* deberá ser modificado.

- Esta carpeta contiene además el archivo "*Makefile.rand*" que sirve para generar los ejecutables y ha sido generado por "rpcgen". Tampoco demanda que se altere.

“rand_server.c” modificado por RPCGEN

```
/*
 * 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 "rand.h"

void * initialize_random_1_svc(long *argp, struct svc_req *rqstp)
{
        static char * result;

    /*
     * insert server code here
    */

    return (void *) &result;
}

double * get_next_random_1_svc(void *argp, struct svc_req *rqstp)
{
    static double  result;

    /*
        * insert server code here
    */

    return &result;
}
```

“rand_server.c” modificado por el programador

```
#include <stdlib.h>
#include "rand.h"

void * initialize_random_1_svc(long *argp, struct svc_req *rqstp)
{
  static char * result;

  srand48(*argp);
  result=(void *)NULL;
  return (void *) &result;
}

double * get_next_random_1_svc(void *argp, struct svc_req *rqstp)
{
  static double  result;

  result=drand48();
  return &result;
}
```

“rand_client.c” modificado por el programador

```
#include <stdlib.h>
#include <stdio.h>
#include "rand.h"

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

  char *host;
  CLIENT *clnt;

  void  *result_1;
  double  *result_2;
  char *get_next_random_1_arg;

  long semilla;
  int i, iters;

  if (argc != 4) {
   fprintf (stderr, "uso: %s servidor semilla iteraciones\n", argv[0]);
   exit (1);
  }

  host = argv[1];
  semilla = (long) atoi(argv[2]);
  iters = atoi(argv[3]);

  clnt = clnt_create (host, RAND_PROG, RAND_VERS, "udp");
  if (clnt == NULL) {
   clnt_pcreateerror (host);
   exit (1);
  }

  result_1 = initialize_random_1(&semilla, clnt);
  if (result_1 == (void *) NULL) {
   clnt_perror (clnt, "call failed");
   exit (2);
  }

  for (i=0; i<iters ;i++) {

   result_2 = get_next_random_1((void*)&get_next_random_1_arg, clnt);
   if (result_2 == (double *) NULL) {
     clnt_perror (clnt, "call failed");
     exit (3);
   } else
     printf("%d: %f\n", i, *result_2);
  }

 clnt_destroy(clnt);
 exit (0);
}
```

Portmapper

- El último paso que se requiere es la instalación del demonio "portmapper/rpcbind"

- Para ello se empleó la orden:

  "**apt-get install portmap**"

  y luego como superusuario se activó el demonio. Esto se hizo en "background" y a través de la línea de comando:

  "**sudo rpcbind &**"