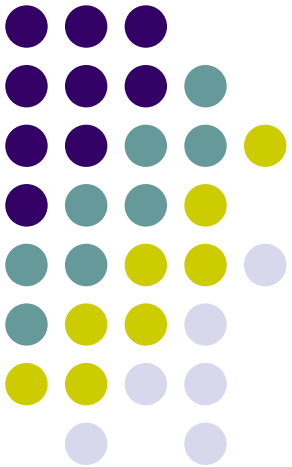
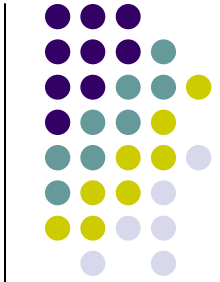
Maestría en Ingeniería de Software
Sistemas Distribuidos

RPC con rpcgen

Hernández González Lizbeth A.
López Martínez María Lina
Navarro Guerrero María Ángeles

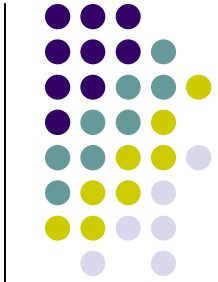


rpcgen



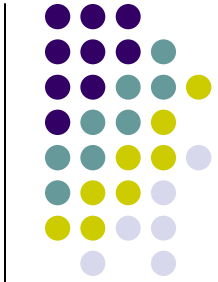
- rpcgen es una herramienta que genera módulos de interfaz de programas remotos.
- Reduce significativamente el tiempo de desarrollo.
- Compila código fuente escrito en RPC Language (RPCL).
- RPCL es similar a C en cuanto a sintaxis y estructura.
- rpcgen produce uno o más módulos fuente en C que serán compilados en un compilador C.

Conversión de llamadas locales a remotas



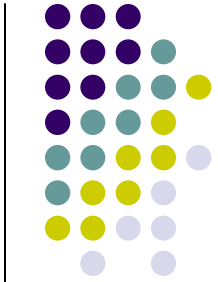
- Crear un archivo de especificación con extensión .x (con RPCL).
- Ejecutar **rpcgen** con opciones -a y -C para generar archivos necesarios.
- Insertar programa que va a llamar a la función en el archivo _client.c generado por rpcgen.
- Insertar código de función local en el archivo _server.c generado por rpcgen.
- Usar el **makefile** generado para compilar los archivos. Es posible detectar errores en las definiciones de tipos en el archivo de especificación.
- Intentar compilar los programas usando el makefile generado.
- Jugar, (fiddle), con el _server.c y el _client.c hasta que trabajen. Este juego puede no ser necesario si solo se utilizan tipos simples de datos.

incremento.x



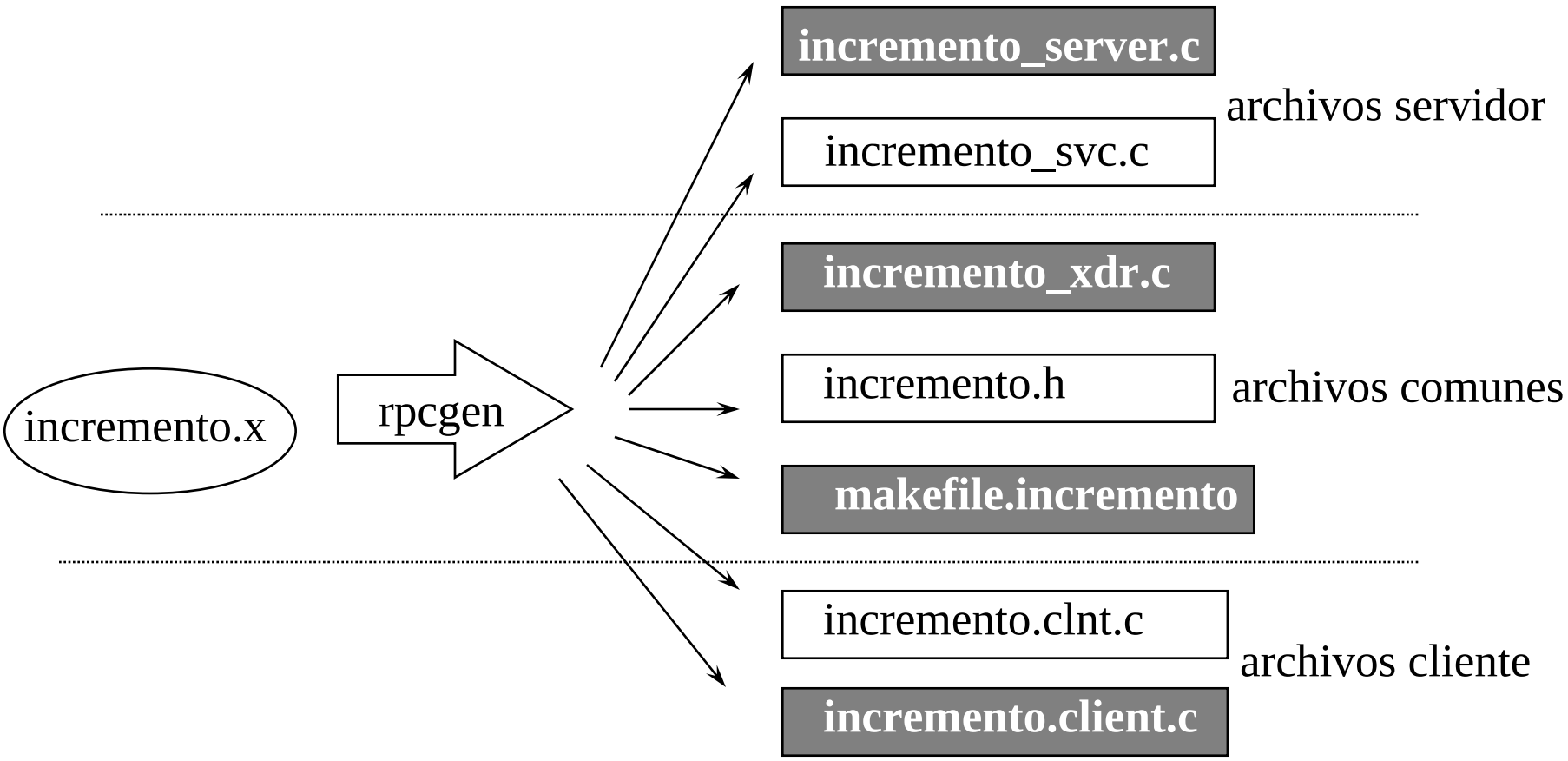
```
/*definicion del programa en RPCL*/
//nombre del programa
program PROGRAMAINCREMENTO {
        //nombre del proceso
    version INCREMENTOVERS {int INCREMENTO(int) = 1;
    } = 1;  //numero de la versión
} = 0x20000001;        //numero del programa
```

Números de programas remotos

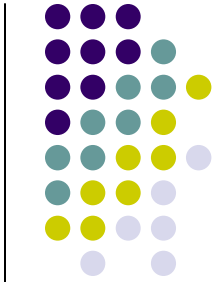


Rango (hexadecimal)	Administrador	Tiempo de vida	Distribución
00000000 - 1FFFFFFF	Sun	Permanente	Pública
20000000 - 3FFFFFFF	Local	Desconocido	Local
40000000 - 5FFFFFFF	Desarrollador	Temporal	Local
50000000 - 7FFFFFFF	Reservado		
80000000 - 9FFFFFFF	Reservado		
A0000000 - BFFFFFFF	Reservado		
C0000000 - DFFFFFFF	Reservado		
E0000000 - FFFFFFFF	Reservado		

Archivos generados por rpcgen



Sintaxis de rpcgen



rpcgen infile

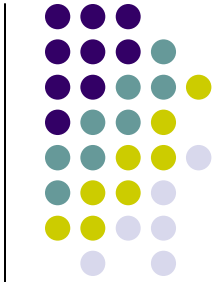
rpcgen [-a] [-A] [-b] [-C] [-D name [= value]]
[-i size] [-I [-K seconds]] [-L]
[-M] [-N] [-T] [-Y pathname] infile

rpcgen [-c | -h | -m | -t | -Sc | - Ss | -Sm]
[-o outfile] [infile]

rpcgen [-s nettype] [-o outfile] [infile]

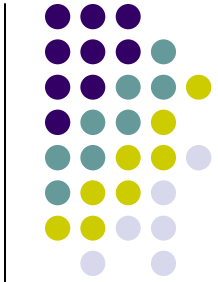
rpcgen [-n netid] [-o outfile] [infile]

Opciones de rpcgen

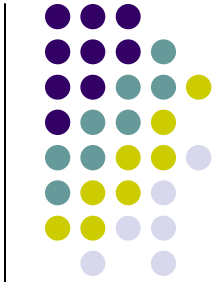


-D*nombre*[=*valor*]	Define un símbolo (igual que #define)
-I	Genera un código para soporte de inted en el servidor (SunOS 4.1)
-K *segundos*	Servidor existe despues de segundos de inactividad
-L	Los errores del servidor seran impresos en el syslog
-T	Genera código para soportar tablas de atención de RPC
-s *transporte*	Genera el código del servidor que soporta el transporte
-o *archivo-salida*	Nombre del archivo de salida
-c	Sólo genera rutinas XDR
-h	Sólo genera el archivo de encabezado
-l	Sólo genera los stubs del cliente
-m	Sólo genera los stuns del servidor
-t	Genera la tabla de atención de RPC
-a	**Genera todos los archivos**
-b	Modo backward de compatilidad (genera código para SunOS4.1)
-C	**Genera lenguaje C de ANSI**
-i *tamaño*	Tamaño a partir del cual se empieza a generar código in-line
-N	Soporta varios argumentos y llamado por valor
-Sc	Genera esqueleto cliente que usa procedimientos remotos
-Ss	Genera esqueleto servidor que define los procedimientos remotos
-Y *path*	Nombre del directorio para encontrar preprocesadores de C (cpp)

Archivos generados por rpcgen



- Comando:
  $rpgen –a –C incremento.x
- Resultado:
  - **Makefile.incremento**
    makefile para compilar el código del cliente y del servidor.
  - **incremento_clnt.c**
    contiene el stub del cliente, el cual usualmente no es modificado.
  - **incremento_svc.c**
    contiene stub servidor
  - **Incremento.h**
    contiene todos los tipos XDR generados a partir de la especificación



- **incremento_client.c**

  programa esqueleto del cliente con llamadas dummy al servicio remoto. Inserta código para asignar los valores de los argumentos para el servicio remoto.

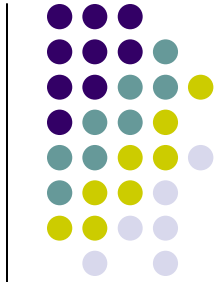
- **incremento_server.c**

  contiene stubs para servicios remotos. Inserta código para la versión local dentro de los stubs.

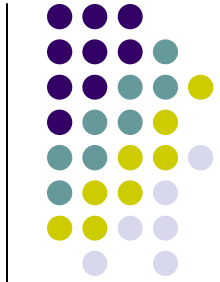
- **incremento_xdr.c**

  contiene filtros XDR necesarios para los stubs del cliente y servidor.

Esqueleto de incremento_client.c

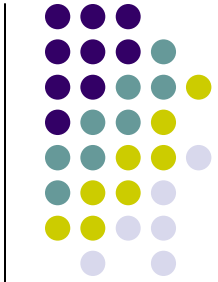


```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */

#include "incremento.h"
void programaincremento_1(char *host)
{
        CLIENT *clnt;
        int  *result_1;
        int  incremento_1_arg;

#ifndef     DEBUG
        clnt = clnt_create (host, PROGRAMAINCREMENTO, INCREMENTOVERS, "udp");
        if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit (1);
        }
#endif      /* DEBUG */
        result_1 = incremento_1(&incremento_1_arg, clnt);
```

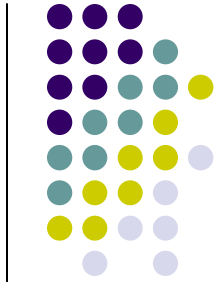


```
if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");

    }
#ifndef     DEBUG
    clnt_destroy (clnt);
#endif       /* DEBUG */
}
int main (int argc, char *argv[])
{
    char *host;
    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    programaincremento_1 (host);
exit (0);
}
```

incremento_client.c FINAL

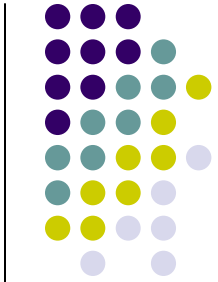


```
#include "incremento.h"

int main(int argc, char  *argv[])
{
      CLIENT *clnt;
      int  *result_1;
      int  incremento_1_arg;
      int  numero;
      char *host;

      if (argc!=3) {
      fprintf(stderr, "Usar: %s host numero\n", argv [0]);
                exit(1);
      }
      host=argv[1];
      clnt = clnt_create (host, PROGRAMAINCREMENTO, INCREMENTOVERS,"udp");
      if (clnt == NULL) {
                clnt_pcreateerror (host);
                exit;
      }
```



```
    numero = (int)atoi(argv[2]);
    result_1 = incremento_1(&numero, clnt);

    if (result_1 == (int *) NULL) {
            clnt_perror (clnt, "la llamada al proc. remoto fallo");
    }

    printf("Resultado = %d\n",*result_1);

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

Esqueleto de incremento_server



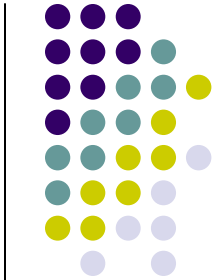
```
/*
 * This is sample code generated by rpcgen.
 * These are only templates and you can use them
 * as a guideline for developing your own functions.
 */
#include "incremento.h"

int * incremento_1_svc(int *argp, struct svc_req *rqstp)
{
    static int  result;
    /*
     * insert server code here
     */
    return &result;
}
```

"_" indica numero de version del proced.

argumento

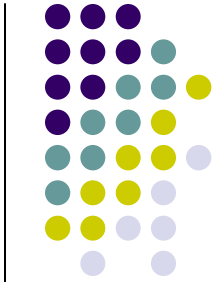
estructura que guarda la información del contexto y la del transporte

incremento_server FINAL



```
/*
 * 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 "incremento.h"

int * incremento_1_svc(int *argp, struct svc_req *rqstp)
{
	static int  result;
	result = *argp+1;
	return &result;
}
```

Compilación

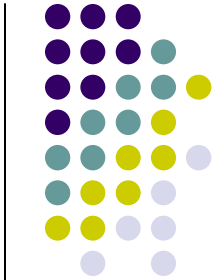


$make –f Makefile.incremento

```
cc -g    -c incremento_clnt.c -o incremento_clnt.o
cc -g    -c incremento_client.c -o incremento_client.o
cc -g     -o incremento_client  incremento_clnt.o incremento_client.o
    -lnsl
cc -g    -c incremento_svc.c -o incremento_svc.o
cc -g    -c incremento_server.c -o incremento_server.o
cc -g     -o incremento_server  incremento_svc.o
          incremento_server.o  -lnsl
```

- Se crean dos archivos ejecutables: incremento_client e incremento_server

Ejecución



- Desde el servidor:

  $ ./incremento_server

  $ ps (debe verse el proceso corriendo)

- Desde el cliente:

  $ ./incremento_client localhost 2

  Resultado = 2