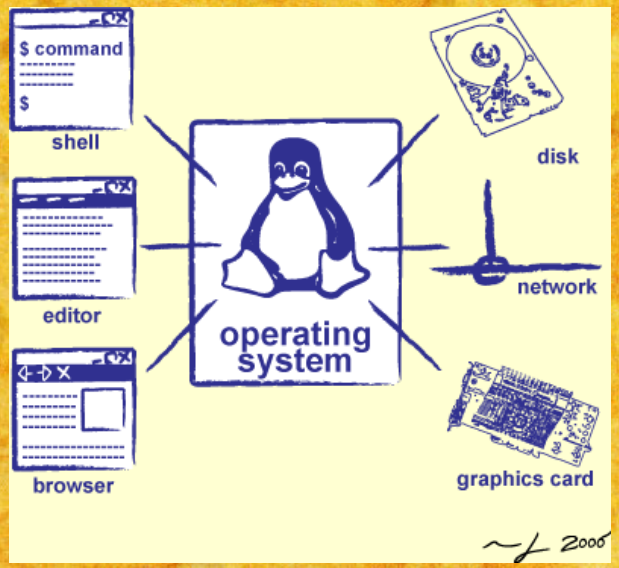
3-1.- Linux RPC


$ command
$
shell
editor
browser
operating system
disk
network
graphics card
2006

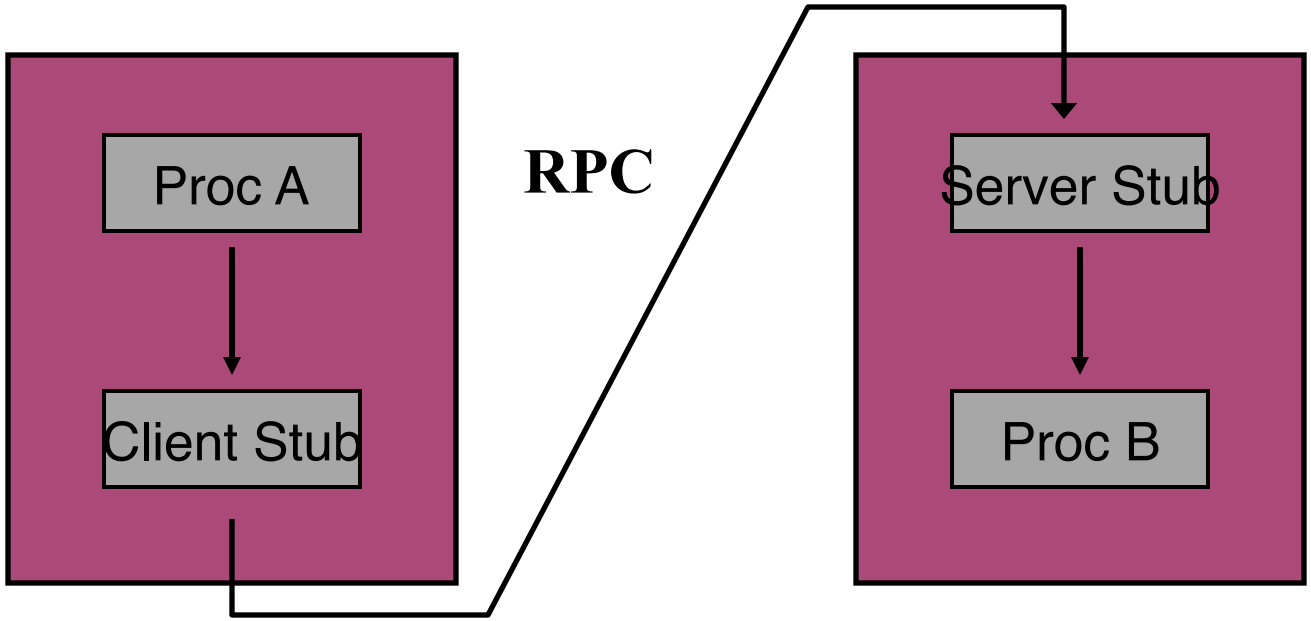
RPC

ONC (Open Network Computing)

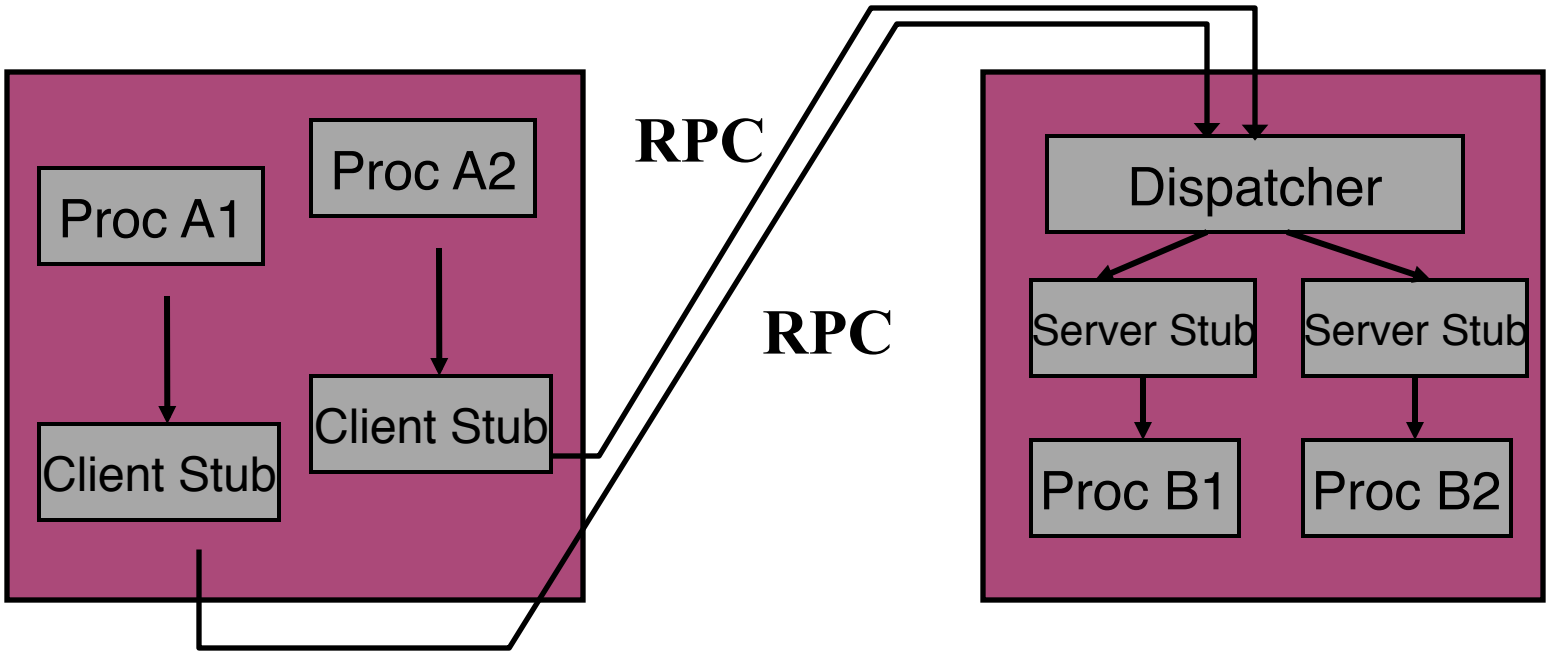
- Rutinas para la conversión de datos XDR
- XDR maneja estructuras complejas
- Herramienta de generación de programas

Proceso de programación RPC

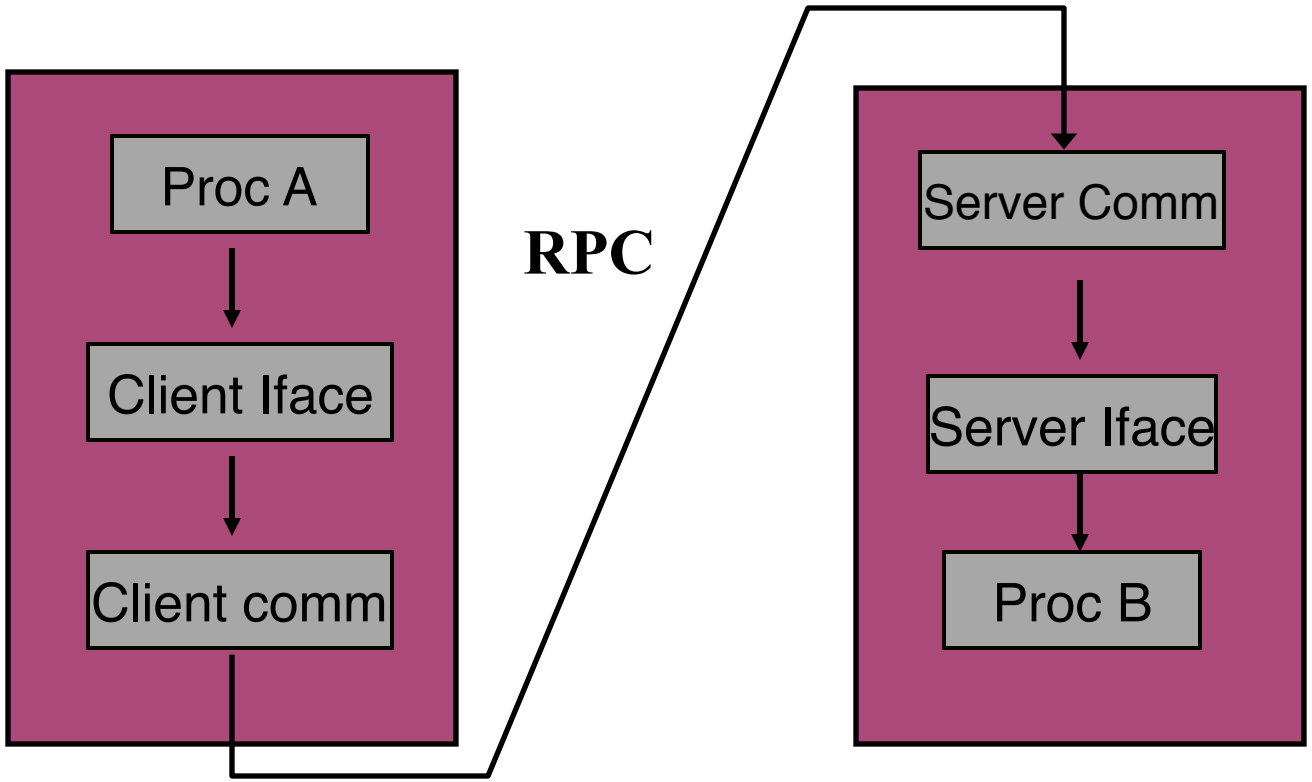
- Programación de métodos locales y remotos.



RPC Dispatching



RPC Interface Specification



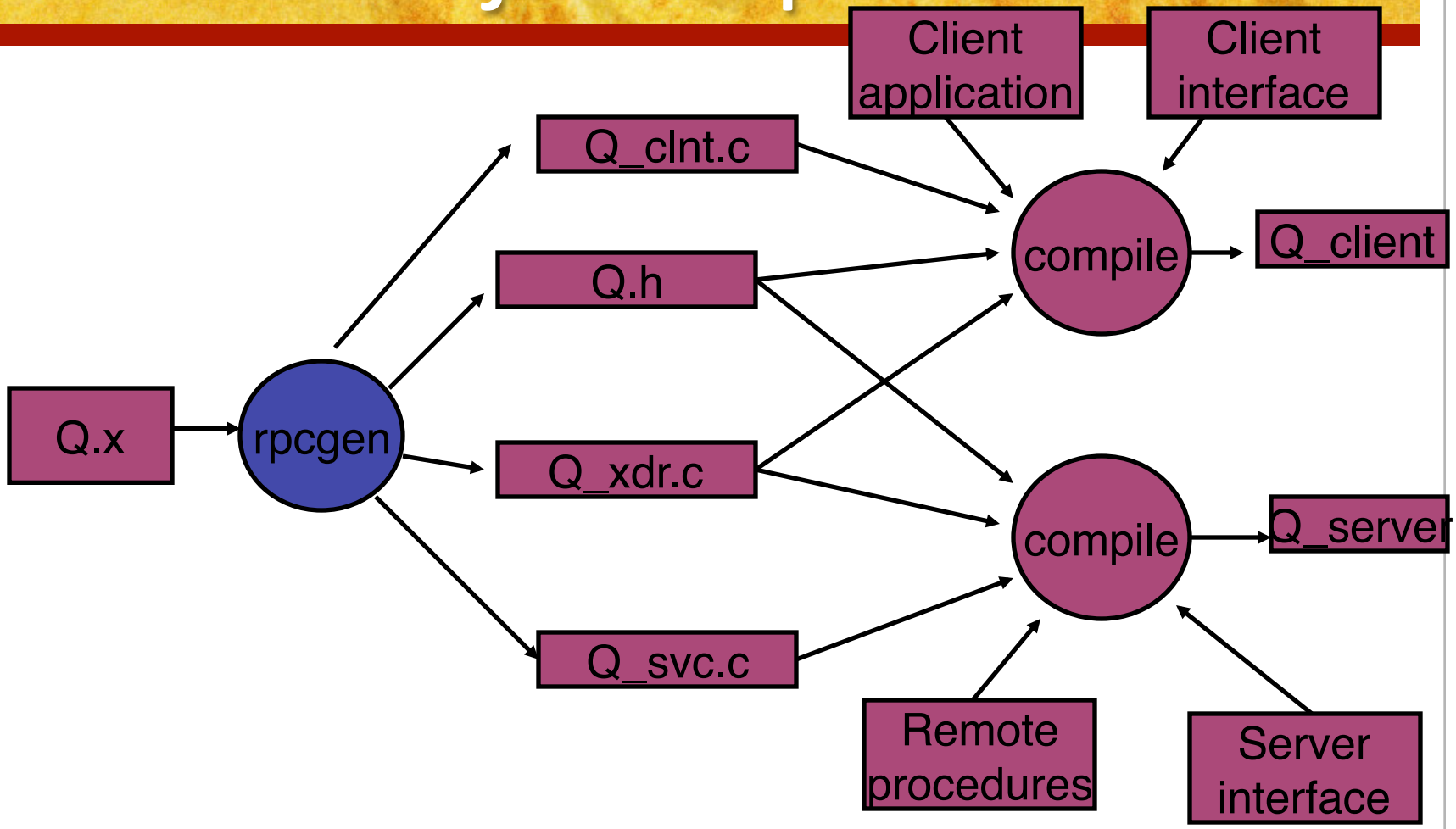
Entrada y salida rpcgen

- Entrada
  - Q.x — Archivo de especificación
- Salida
  - Q.h — Archivo headers
  - Q_xdr.c — Procedimientos XDR para hacer marshaling
  - Q_clnt.c — stub de comunicación del lado del cliente
  - Q_svc.c — stub de comunicación del lado del servidor

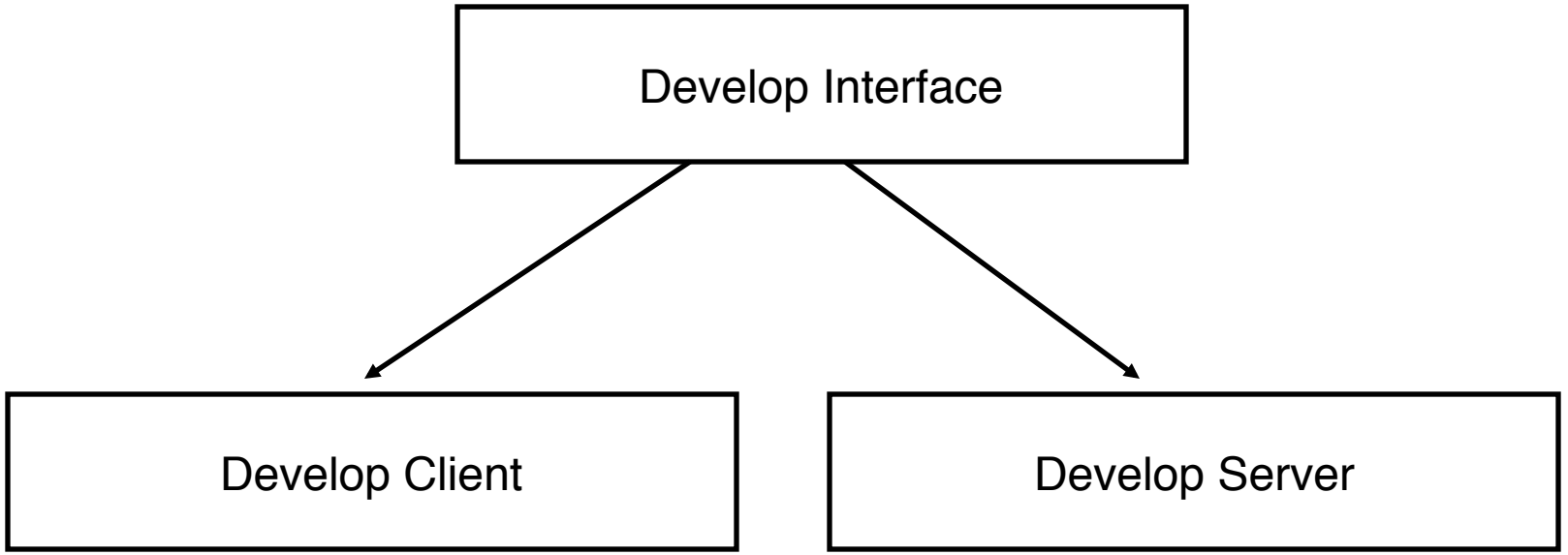
Entrada y salida rpcgen

- Salida
  - Q_client.c Plantilla para invocar los procedimientos remotos
  - Q_serve.c Plantilla para implemetar los procedimientos remotos

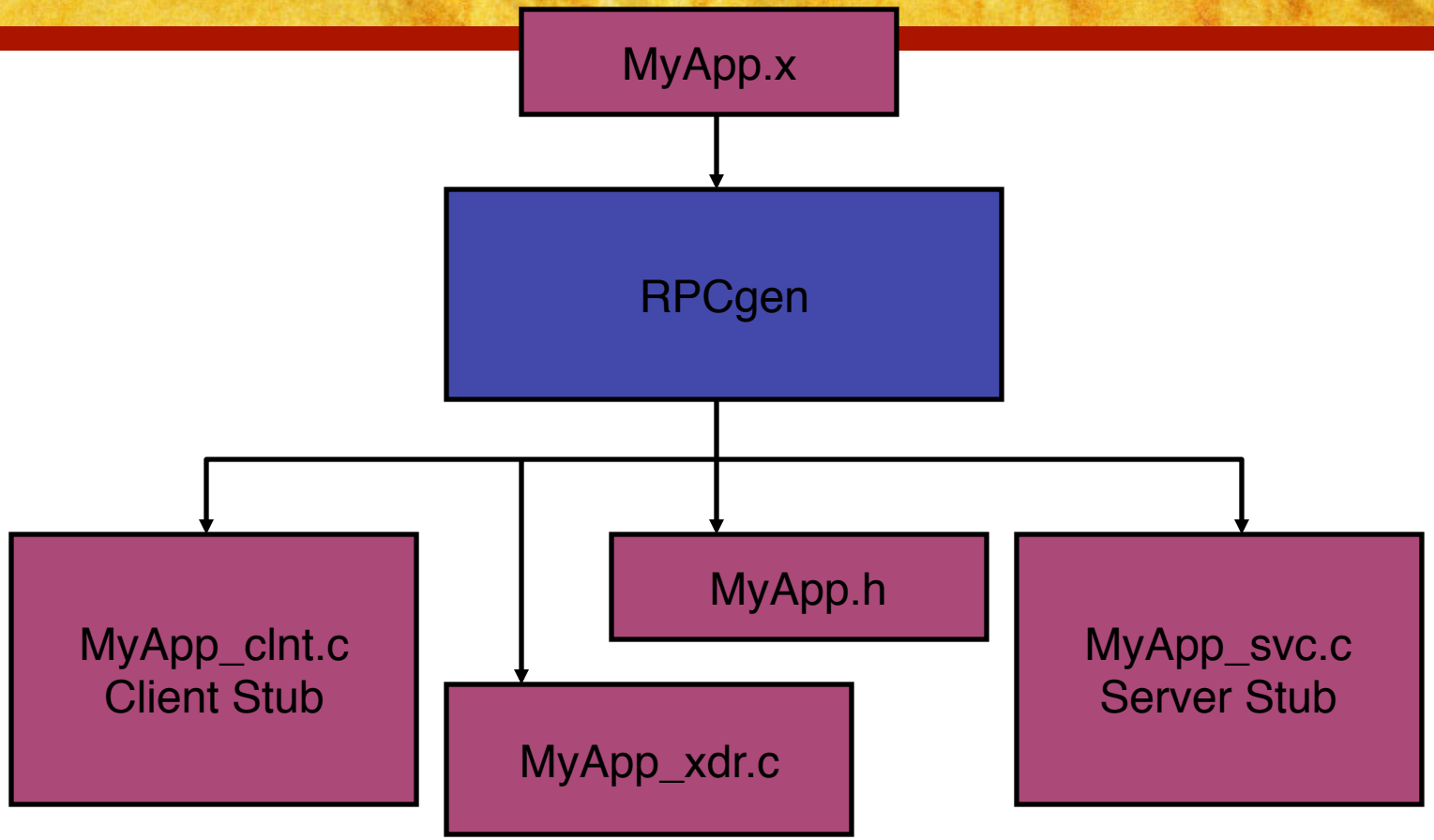
Flujo del proceso RPC



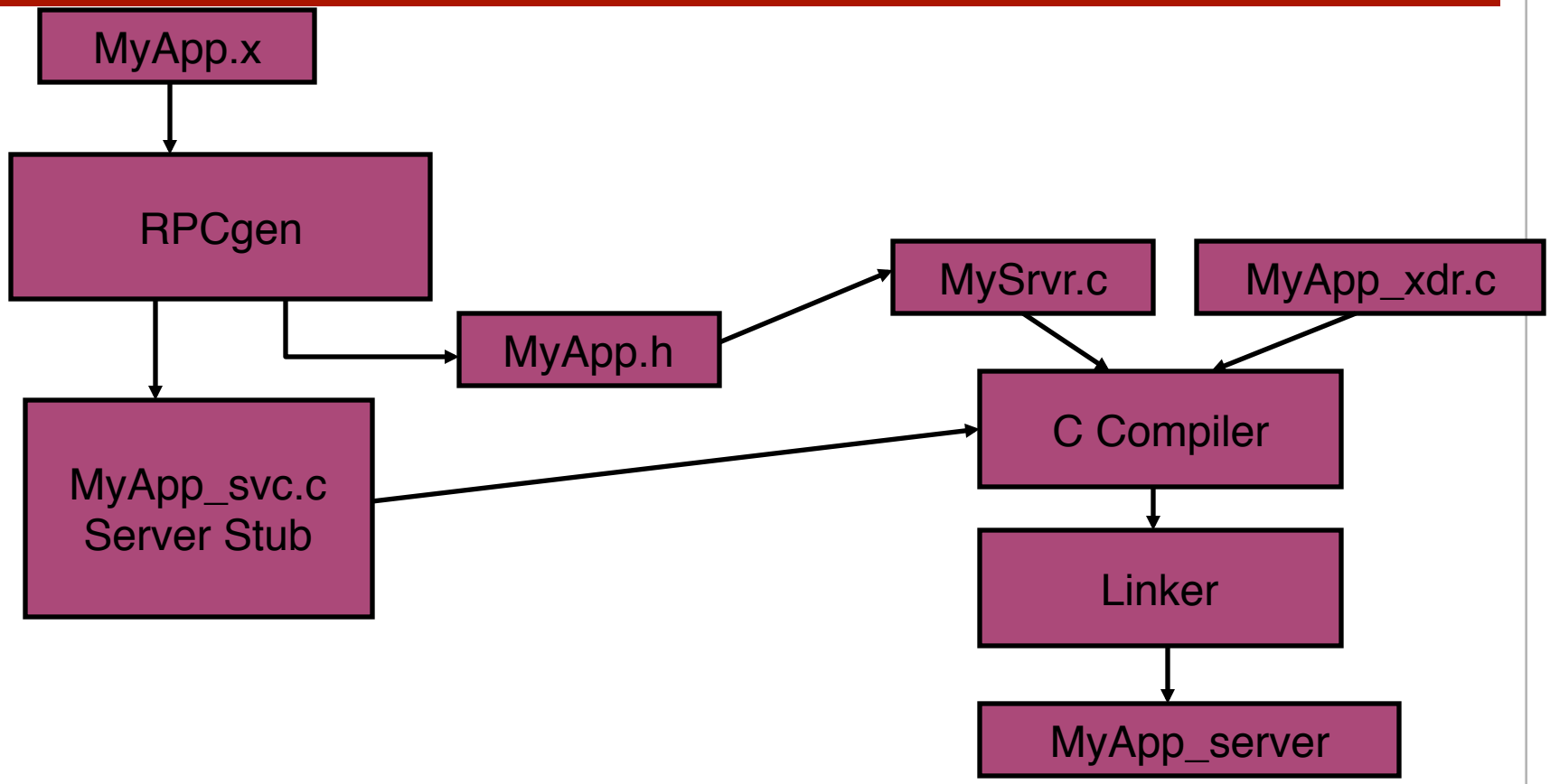
Proceso de desarrollo RPC



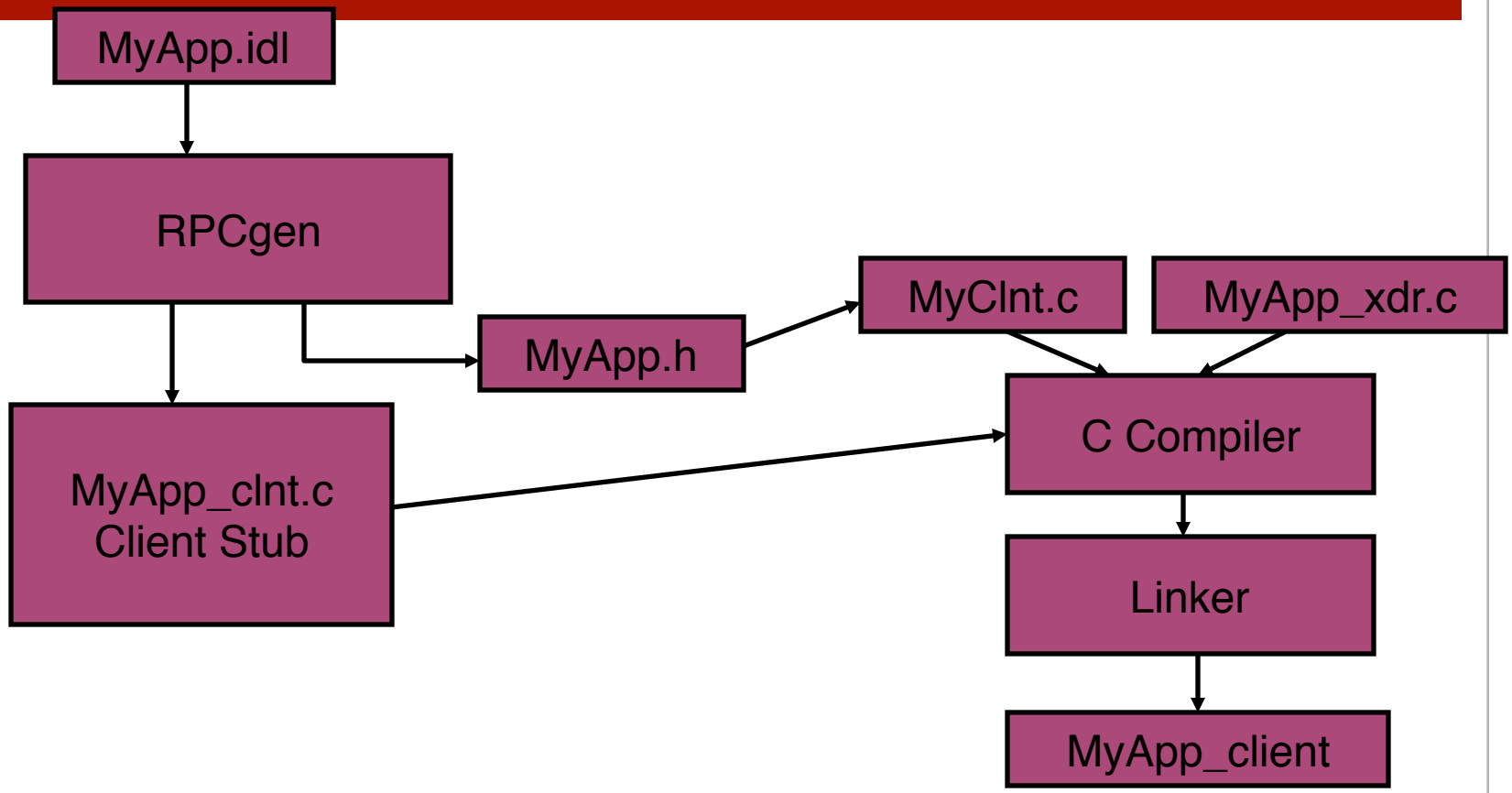
Desarrollo de la Interface



Developing the Server



Developing the Client



Proceso de conexión

- Portmap debe estar corriendo (pruebe rpcinfo)
- Crea un servicio UDP / TCP
- Portmap regista el servcio UDP/TCP
- Ejecuta el servicio...

Registar el servicio

```
svc_register(port#, SERV#, VER#, serv_func, proto#);
```

port#:Puerto en el que el servicio está activo

SERV#: identificador único para el servicio

VER#: versión del servicio

serv_func: función a ejecutar del servicio

proto#: IPPROTO_UDP o IPPROTO_TCP

Como se conecta el cliente

- Hacer una llamada a un procedimiento remoto
- Encontrar el host remoto
- Determinar el puerto del servicio (usando Portmap)
- Crear una conexión al proceso en el servidor

Como se conecta el cliente

```
clnt_create(server, SERV#, VER#, proto#);

remote procedure call...

Clnt_destroy(CLIENT);
```


Ejemplo

- Sumar dos números
  - Pasar dos párametros enterso
  - UDP / IP port connection
- Archivos fuentes:
  - suma.x
  - suma_client.c // a Modificar
  - suma_server.c // aModificar

suma.x

```
struct numeros{
    int a;
    int b;
};

program ADICIONPROG{
    version ADDITIONVERS{
        int SUMA(numeros)=1;
    }=1;
}= 0x2fffffff;
```

Compilar y ejecutar

- make –f M akefile.suma
- Servidor
  - sudo ./suma_server
- Cliente
  - ./suma_client
- parece que no hace nada o hace nada

Agregar un mensaje

- Modificar la plantilla suma_server.c

```
#include "suma.h"

int *suma_1_svc(numeros *argp, struct svc_req *rqstp){
        static int  result;
        /*
         * insert server code here
         */
        printf( "Peticion acpetada\n ");
        return &result;
}
```

- Volver a compilar/ejecutar observar que ocurre

suma_client.c

- Cómo se pueden pasar parámetros al servidor?

suma_client.c

- Cómo se pueden pasar parámetros al servidor?
- Modificar de

```
adicionprog_1(char *host){
        CLIENT *clnt;
        int  *result_1;
        numeros  suma_1_arg;
```

a

```
adicionprog_1(char *host, int a, int b){
        CLIENT *clnt;
        int  *result_1;
        numeros  suma_1_arg;
        suma_1_arg.a=a;
        suma_1_arg.b=b; ...
```

- Hacer los cambios necesarios en el main

server_client.c

- Cómo acceder los parámetros desde el servidor?

server_client.c

- Cómo acceder los parámetros desde el servidor?

```
int *  suma_1_svc(numeros *argp, struct svc_req *rqstp){
        static int  result;
        int a= argp->a, b=argp->b;
        printf("Tu quieres sumar los numeros a=%d, b=%d\n", a,b);
        result=argp->a+argp->b;
        return &result;
}
```

- Cómo se imprimiría el resultado del lado del cliente?

Remarks

- Los procedimientos remotos utilizan punteros como argumentos.
- Generalmente regresan punteros
- En general los procedimientos remotos son nombrados utilizando el nombre de la función en minúsculas y agregando "_" y el número de versión (v.g SUMA -> suma_1)
- RPC language (el utilizado en Q.x) soporta, definition, enumeration, union, enumerations, typedef y Constants

Reglas de definición programs

PROGRAM_DEFINITION:
program <program-ident> {
VERSION-LIST
} = valor;

VERSION-LIST:
VERSION ; | VERSION ; VERSION-LIST

VERSION:
version <version-id> {
PROCEDURE-LIST
} = valor;

PROCEDURE-LIST:
PROCEDURE ; | PROCEDURE-LIST

PROCEDURE:
<tipo> <procedure-ident> (<tipo>) = valor;

```
/* Calculadora RPC */

program  RPCCAL{
   version RPCCALVERS{
         int SUMA(numeros)=1;
         int RESTA(numeros)=2;
         int PRODUCTO(numeros)=3;
   }=1;
}=44;
```