

# Remote Procedure Call

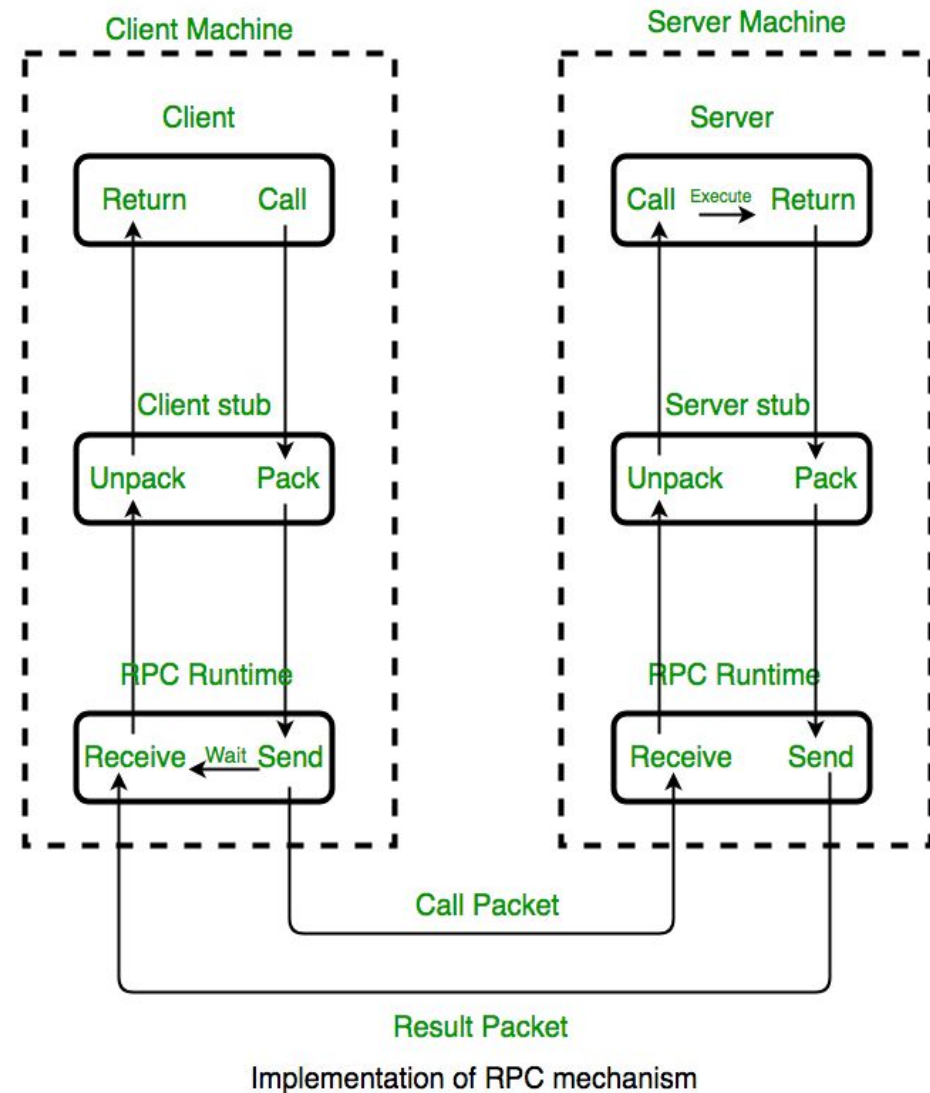
# Remote Procedure Call

- A remote procedure call is an **interprocess communication** technique that is used for client-server based applications.
- It is also known as a **subroutine call** or a **function call**.



# Implementing RPC Mechanism

- Client procedure **calls** the client stub in a normal way.
- Client stub **builds a message** and **traps** to the kernel.
- Kernel **sends** the message to remote kernel.
- Remote kernel **gives** the message to server stub.
- Server stub **unpacks** parameters and **calls** the server.
- Server **computes** results and **returns** it to server stub.
- Server stub **packs** results in a message to client and **traps** to kernel.
- Remote kernel **sends** message to client kernel.
- Client kernel **gives** message to client stub.
- Client stub **unpacks** results and **returns** to client.



# RPCGEN

- **RPCGEN** is an interface generator pre-compiler for Sun Microsystems ONC RPC (Open Network Computing RPC).
- It creates client and server stubs in C based on information contained within a file called IDL (interface definition file) file. This file is written in a language called RPCL – remote procedure call language.
- **Client side program** to get numbers from the user, to call the relevant method in the server side and to send those numbers to that method in the server side.
- **Server side program** to do the addition and to send back the result to the client side.

Let the server procedure run as the localhost for this example.

# IDL

- An IDL is a file (suffixed with .x) which optionally begins with a bunch of type definitions and then defines the remote procedures.
- A set of remote procedures are grouped into a *version*.
- One or more versions are grouped into a *program*.

# STEPS

- Install **rpcbind** in your machine  
`sudo apt-get install rpcbind`

```
psg@psg-OptiPlex-3060:~$ rpcinfo
Command 'rpcinfo' not found, but can be installed with:

sudo apt install rpcbind

psg@psg-OptiPlex-3060:~$ sudo apt-get install rpcbind
[sudo] password for psg:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libtirpc1
The following NEW packages will be installed:
  libtirpc1 rpcbind
0 upgraded, 2 newly installed, 0 to remove and 323 not upgraded.
Need to get 118 kB of archives.
After this operation, 364 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libtirpc1 amd64 0.2.5-1.2ubuntu0.1 [75.7 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu bionic-updates/main amd64 rpcbind amd64 0.2.3-0.6ubuntu0.18.04.4 [42.1 kB]
Fetched 118 kB in 1s (161 kB/s)
Selecting previously unselected package libtirpc1:amd64.
(Reading database ... 172438 files and directories currently installed.)
Preparing to unpack .../libtirpc1_0.2.5-1.2ubuntu0.1_amd64.deb ...
Unpacking libtirpc1:amd64 (0.2.5-1.2ubuntu0.1) ...
Selecting previously unselected package rpcbind.
Preparing to unpack .../rpcbind_0.2.3-0.6ubuntu0.18.04.4_amd64.deb ...
Unpacking rpcbind (0.2.3-0.6ubuntu0.18.04.4) ...
Processing triggers for ureadahead (0.100.0-20) ...
ureadahead will be reprofiled on next reboot
Processing triggers for libc-bin (2.27-3ubuntu1.2) ...
Setting up libtirpc1:amd64 (0.2.5-1.2ubuntu0.1) ...
Processing triggers for systemd (237-3ubuntu10.50) ...
Processing triggers for man-db (2.8.3-2) ...
Setting up rpcbind (0.2.3-0.6ubuntu0.18.04.4) ...
Created symlink /etc/systemd/system/multi-user.target.wants/rpcbind.service → /lib/systemd/system/rpcbind.service.
Created symlink /etc/systemd/system/sockets.target.wants/rpcbind.socket → /lib/systemd/system/rpcbind.socket.
Processing triggers for libc-bin (2.27-3ubuntu1.2) ...
Processing triggers for systemd (237-3ubuntu10.50) ...
Processing triggers for ureadahead (0.100.0-20) ...
```



# STEPS

- rpcinfo

```
psg@psg-OptiPlex-3060:~$ rpcinfo
  program version netid  address                service  owner
  100000    4      tcp6   :::0.111               portmapper superuser
  100000    3      tcp6   :::0.111               portmapper superuser
  100000    4      udp6   :::0.111               portmapper superuser
  100000    3      udp6   :::0.111               portmapper superuser
  100000    4      tcp    0.0.0.0.0.111         portmapper superuser
  100000    3      tcp    0.0.0.0.0.111         portmapper superuser
  100000    2      tcp    0.0.0.0.0.111         portmapper superuser
  100000    4      udp    0.0.0.0.0.111         portmapper superuser
  100000    3      udp    0.0.0.0.0.111         portmapper superuser
  100000    2      udp    0.0.0.0.0.111         portmapper superuser
  100000    4      local  /run/rpcbind.sock     portmapper superuser
  100000    3      local  /run/rpcbind.sock     portmapper superuser
```



# add.x

This is the IDL file

```
/*combine the arguments to be passed to the server side in a structure*/
```

```
struct numbers{
```

```
    int a;
```

```
    int b;
```

```
};
```

```
program ADD_PROG{
```

```
version ADD_VERS{
```

```
    int add(numbers)=1;
```

```
    }=1;
```

```
    }=0x4562877;
```

# add.x

- numbers -> name of the structure which sends the parameters to the server
- ADD\_PROG -> name of the program
- ADD\_VERS -> name of the program version
- add(numbers) -> method that we are going to call remotely. We pass the numbers structure as the parameter to this method.

- Compile this IDL file by using the following command.  
**rpcgen -a -C add.x**
- Should execute this command in the directory where the idl file resides.
- '-C ' in the command is for C language
- '-a' is to generate all the files including samples.

It will generate the following files.

add.h -> header file

add\_client.c -> client program

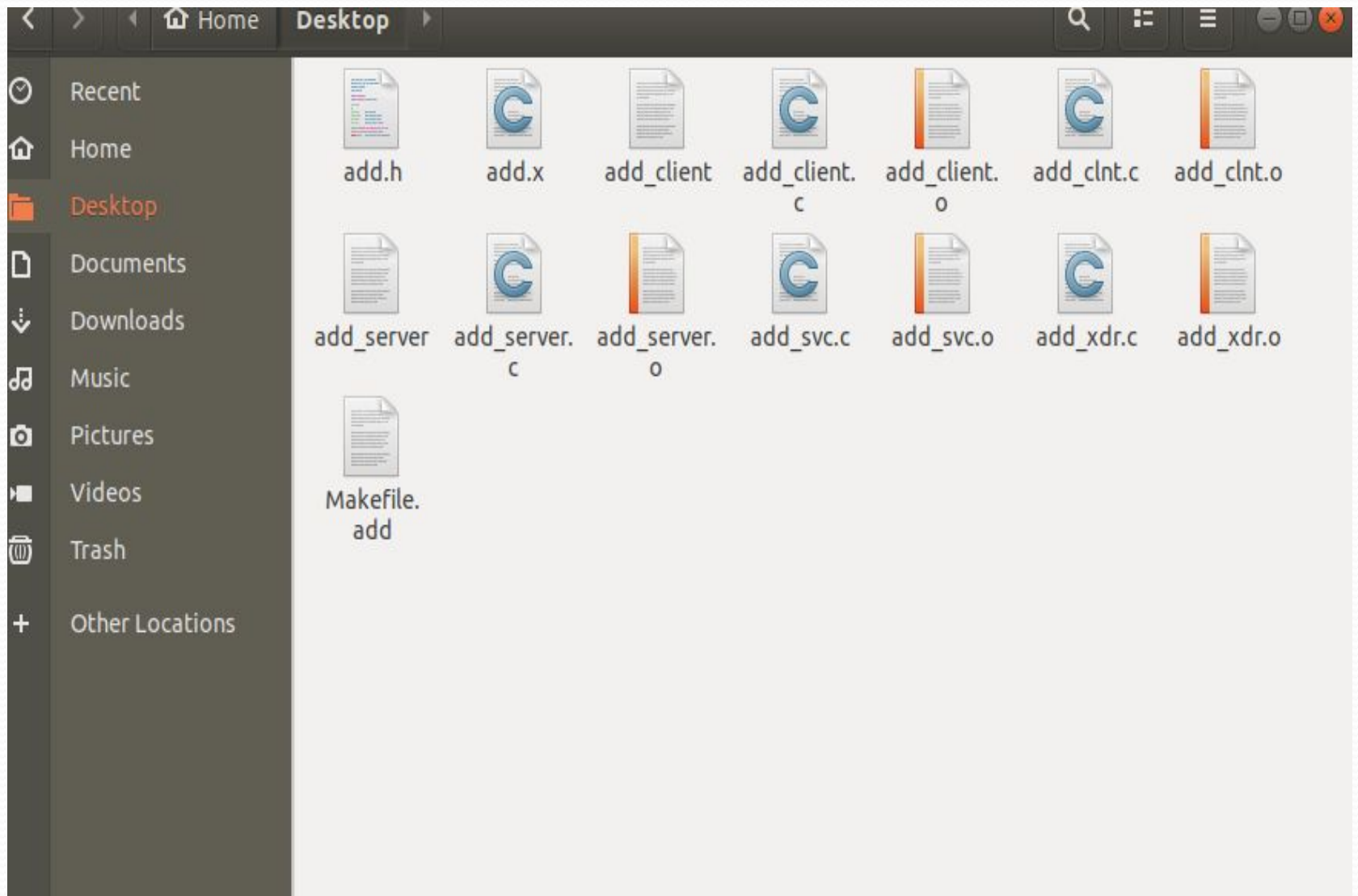
add\_clnt.c -> client stub

add\_server.c -> server program

add\_svc.c -> server skeleton

add\_xdr.c -> XDR routines used by both the client and the server

Makefile.add -> Makefile



# Edit add\_server.c

```
#include "add.h"
int *
add_1_svc(numbers *argp, struct svc_req *rqstp)
{
    static int result;
    printf("add(%d,%d) is called\n", argp->a,argp->b);
    result = argp->a + argp->b;
    return &result;
}
```

# Edit add\_client.c

```
#include "add.h"
void
add_prog_1(char *host,int x,int y)
{
    CLIENT *clnt;
    int *result_1;
    numbers add_1_arg;
    #ifndef DEBUG
    clnt = clnt_create (host, ADD_PROG, ADD_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    } #endif /* DEBUG */
```

# Edit add\_client.c

```
add_1_arg.a=x;
add_1_arg.b=y;
result_1 = add_1(&add_1_arg, clnt);
if (result_1 == (int *) NULL) {
    clnt_perror (clnt, "call failed");
} else{
    printf("Result:%d\n", *result_1 );
}
#ifdef DEBUG
clnt_destroy (clnt);
#endif /* DEBUG */ }
```



# Edit add\_client.c

```
int
main (int argc, char *argv[])
{
    char *host;
    if (argc < 4) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1); }
    host = argv[1];
    add_prog_1 (host,atoi(argv[2]),atoi(argv[3]));
    exit (0);
}
```

- Since we've changed those two files now we have to compile all the files again. Use the following command to do that. It will generate all the **object files**.

**make -f Makefile.add**

- Now open up two terminals and run the server in a one and client in the other.
- To start server --> `:-$ sudo ./add_server`
- To start client --> `:-$ sudo ./add_client localhost 5 8`

```
psg@psg-OptiPlex-3060:~$ cd Desktop
psg@psg-OptiPlex-3060:~/Desktop$ rpcgen -a -C add.x
psg@psg-OptiPlex-3060:~/Desktop$ make -f Makefile.add
cc -g      -c -o add_clnt.o add_clnt.c
cc -g      -c -o add_client.o add_client.c
cc -g      -c -o add_xdr.o add_xdr.c
cc -g      -o add_client  add_clnt.o add_client.o add_xdr.o -lnsl
cc -g      -c -o add_svc.o add_svc.c
cc -g      -c -o add_server.o add_server.c
cc -g      -o add_server  add_svc.o add_server.o add_xdr.o -lnsl
psg@psg-OptiPlex-3060:~/Desktop$ sudo ./add_server
add(1,2) is called
```

```
psg@psg-OptiPlex-3060:~/Desktop$ sudo ./add_client localhost 1 2
[sudo] password for psg:
Result:3
psg@psg-OptiPlex-3060:~/Desktop$
```