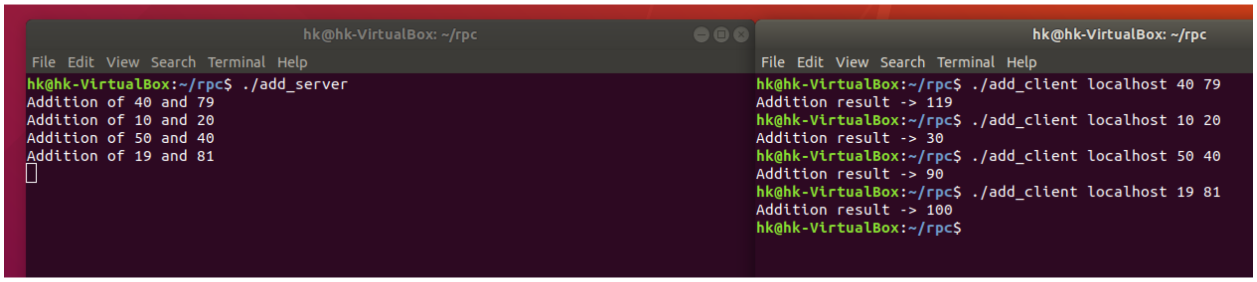
CN Assignment

RPC (Add two numbers)

-By Himanshu Kwatra(187121)


hk@hk-VirtualBox: ~/rpc
File Edit View Search Terminal Help
hk@hk-VirtualBox: ~/rpc
File Edit View Search Terminal Help

```
hk@hk-VirtualBox:~/rpc$ ./add_server
Addition of 40 and 79
Addition of 10 and 20
Addition of 50 and 40
Addition of 19 and 81
```

```
hk@hk-VirtualBox:~/rpc$ ./add_client localhost 40 79
Addition result -> 119
hk@hk-VirtualBox:~/rpc$ ./add_client localhost 10 20
Addition result -> 30
hk@hk-VirtualBox:~/rpc$ ./add_client localhost 50 40
Addition result -> 90
hk@hk-VirtualBox:~/rpc$ ./add_client localhost 19 81
Addition result -> 100
hk@hk-VirtualBox:~/rpc$
```

Code-

add_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 "add.h"

int *
add_1_svc(numbers *argp, struct svc_req *rqstp)
{
	static int  result;

	printf("Addition of %d and %d\n",argp->a,argp->b);
	result = argp->a + argp->b;

	return &result;
}
```

add_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 "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 */
	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("Addition result -> %d\n",*result_1);
	}
#ifndef DEBUG
	clnt_destroy (clnt);
#endif   /* DEBUG */
}


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

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

add_clnt.c

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#include <memory.h> /* for memset */
#include "add.h"

/* Default timeout can be changed using clnt_control() */
static struct timeval TIMEOUT = { 25, 0 };

int *
add_1(numbers *argp, CLIENT *clnt)
{
	static int clnt_res;
	memset((char *)&clnt_res, 0, sizeof(clnt_res));
	if (clnt_call (clnt, add,
		(xdrproc_t) xdr_numbers, (caddr_t) argp,
		(xdrproc_t) xdr_int, (caddr_t) &clnt_res,
		TIMEOUT) != RPC_SUCCESS) {
		return (NULL);
	}
	return (&clnt_res);
}
```

add_svc.c

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#include "add.h"
#include <stdio.h>
#include <stdlib.h>
#include <rpc/pmap_clnt.h>
#include <string.h>
#include <memory.h>
#include <sys/socket.h>
#include <netinet/in.h>

#ifndef SIG_PF
#define SIG_PF void(*)(int)
#endif

static void
add_prog_1(struct svc_req *rqstp, register SVCXPRT *transp)
{
	union {
		numbers add_1_arg;
	} argument;
	char *result;
	xdrproc_t _xdr_argument, _xdr_result;
	char *(*local)(char *, struct svc_req *);

	switch (rqstp->rq_proc) {
	case NULLPROC:
		(void) svc_sendreply (transp, (xdrproc_t) xdr_void, (char *)NULL);
		return;

	case add:
		_xdr_argument = (xdrproc_t) xdr_numbers;
		_xdr_result = (xdrproc_t) xdr_int;
		local = (char ()(char *, struct svc_req *)) add_1_svc;
		break;

	default:
		svcerr_noproc (transp);
		return;
	}
	memset ((char *)&argument, 0, sizeof (argument));
	if (!svc_getargs (transp, (xdrproc_t) _xdr_argument, (caddr_t) &argument)) {
		svcerr_decode (transp);
		return;
	}
	result = (*local)((char *)&argument, rqstp);
	if (result != NULL && !svc_sendreply(transp, (xdrproc_t) _xdr_result, result)) {
		svcerr_systemerr (transp);
	}
	if (!svc_freeargs (transp, (xdrproc_t) _xdr_argument, (caddr_t) &argument)) {
		fprintf (stderr, "%s", "unable to free arguments");
		exit (1);
	}
	return;
}

int
main (int argc, char **argv)
{
	register SVCXPRT *transp;

	pmap_unset (ADD_PROG, ADD_VERS);

	transp = svcudp_create(RPC_ANYSOCK);
	if (transp == NULL) {
			fprintf (stderr, "%s", "cannot create udp service.");
			exit(1);
		}
		if (!svc_register(transp, ADD_PROG, ADD_VERS, add_prog_1, IPPROTO_UDP)) {
			fprintf (stderr, "%s", "unable to register (ADD_PROG, ADD_VERS, udp).");
			exit(1);
		}

		transp = svctcp_create(RPC_ANYSOCK, 0, 0);
		if (transp == NULL) {
			fprintf (stderr, "%s", "cannot create tcp service.");
			exit(1);
		}
		if (!svc_register(transp, ADD_PROG, ADD_VERS, add_prog_1, IPPROTO_TCP)) {
			fprintf (stderr, "%s", "unable to register (ADD_PROG, ADD_VERS, tcp).");
			exit(1);
		}

		svc_run ();
		fprintf (stderr, "%s", "svc_run returned");
		exit (1);
		/* NOTREACHED */
}
```

add_xdr.c

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#include "add.h"

bool_t
xdr_numbers (XDR *xdrs, numbers *objp)
{
	register int32_t *buf;

	 if (!xdr_int (xdrs, &objp->a))
		 return FALSE;
	 if (!xdr_int (xdrs, &objp->b))
		 return FALSE;
	return TRUE;
}
```

add.x

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

program ADD_PROG
{
	version ADD_VERS
	{
		int add(numbers)=1;
	}=1;
}=0x23831111;
```

add.h

```
/*
 * Please do not edit this file.
 * It was generated using rpcgen.
 */

#ifndef _ADD_H_RPCGEN
#define _ADD_H_RPCGEN

#include <rpc/rpc.h>


#ifdef __cplusplus
extern "C" {
#endif


struct numbers {
	int a;
	int b;
};
typedef struct numbers numbers;

#define ADD_PROG 0x23831111
#define ADD_VERS 1

#if defined(_STDC) || defined(_cplusplus)
#define add 1
extern  int * add_1(numbers *, CLIENT *);
extern  int * add_1_svc(numbers *, struct svc_req *);
extern int add_prog_1_freeresult (SVCXPRT *, xdrproc_t, caddr_t);

#else /* K&R C */
#define add 1
extern  int * add_1();
extern  int * add_1_svc();
extern int add_prog_1_freeresult ();
#endif /* K&R C */

/* the xdr functions */

#if defined(_STDC) || defined(_cplusplus)
extern  bool_t xdr_numbers (XDR , numbers);

#else /* K&R C */
extern bool_t xdr_numbers ();
#endif /* K&R C */

#ifdef __cplusplus
}
#endif

#endif /* !_ADD_H_RPCGEN */
```

Makefile.add

```
# This is a template Makefile generated by rpcgen

# Parameters

CLIENT = add_client
SERVER = add_server

SOURCES_CLNT.c =
SOURCES_CLNT.h =
SOURCES_SVC.c =
SOURCES_SVC.h =
SOURCES.x = add.x

TARGETS_SVC.c = add_svc.c add_server.c add_xdr.c
TARGETS_CLNT.c = add_clnt.c add_client.c add_xdr.c
TARGETS = add.h add_xdr.c add_clnt.c add_svc.c add_client.c add_server.c

OBJECTS_CLNT = $(SOURCES_CLNT.c:%.c=%.o) $(TARGETS_CLNT.c:%.c=%.o)
OBJECTS_SVC = $(SOURCES_SVC.c:%.c=%.o) $(TARGETS_SVC.c:%.c=%.o)
# Compiler flags

CFLAGS += -g
LDLIBS += -lnsl
RPCGENFLAGS =

# Targets

all : $(CLIENT) $(SERVER)
```

```
$(TARGETS) : $(SOURCES.x)
        rpcgen $(RPCGENFLAGS) $(SOURCES.x)

$(OBJECTS_CLNT) : $(SOURCES_CLNT.c) $(SOURCES_CLNT.h) $(TARGETS_CLNT.c)

$(OBJECTS_SVC) : $(SOURCES_SVC.c) $(SOURCES_SVC.h) $(TARGETS_SVC.c)

$(CLIENT) : $(OBJECTS_CLNT)
        $(LINK.c) -o $(CLIENT) $(OBJECTS_CLNT) $(LDLIBS)

$(SERVER) : $(OBJECTS_SVC)
        $(LINK.c) -o $(SERVER) $(OBJECTS_SVC) $(LDLIBS)

 clean:
         $(RM) core $(TARGETS) $(OBJECTS_CLNT) $(OBJECTS_SVC) $(CLIENT)
$(SERVER)
```