# DISTRIBUTED SYSTEMS AND COMPUTING

## ASSIGNMENT 3

Submitted by:
Shiv Kumar
2016UIT2563

# Practical : Implement Remote Procedure Call using RPCs between client and server

**Code:**

**msg.h**

```
#ifndef _MSG_H_RPCGEN
#define _MSG_H_RPCGEN

#include <rpc/rpc.h>


#ifdef __cplusplus
extern "C" {
#endif


#define MESSAGEPROG 0x20000001
#define PRINTMESSAGEVERS 1

#if defined(__STDC__) || defined(__cplusplus)
#define PRINTMESSAGE 1
extern  int * printmessage_1(char **, CLIENT *);
extern  int * printmessage_1_svc(char **, struct svc_req *);
extern int messageprog_1_freeresult (SVCXPRT *, xdrproc_t, caddr_t);

#else /* K&R C */
#define PRINTMESSAGE 1
extern  int * printmessage_1();
extern  int * printmessage_1_svc();
extern int messageprog_1_freeresult ();
#endif /* K&R C */

#ifdef __cplusplus
}
#endif

#endif /* !_MSG_H_RPCGEN */
```

**msg_svc.c**

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

#include "msg.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
messageprog_1(struct svc_req *rqstp, register SVCXPRT *transp)
{
	union {
		char *printmessage_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 PRINTMESSAGE:
		_xdr_argument = (xdrproc_t) xdr_wrapstring;
		_xdr_result = (xdrproc_t) xdr_int;
		local = (char *(*)(char *, struct svc_req *)) printmessage_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 (MESSAGEPROG, PRINTMESSAGEVERS);

	transp = svcudp_create(RPC_ANYSOCK);
	if (transp == NULL) {
		fprintf (stderr, "%s", "cannot create udp service.");
		exit(1);
	}
	if (!svc_register(transp, MESSAGEPROG, PRINTMESSAGEVERS, messageprog_1,
IPPROTO_UDP)) {
		fprintf (stderr, "%s", "unable to register (MESSAGEPROG,
PRINTMESSAGEVERS, 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, MESSAGEPROG, PRINTMESSAGEVERS, messageprog_1,
IPPROTO_TCP)) {
		fprintf (stderr, "%s", "unable to register (MESSAGEPROG,
PRINTMESSAGEVERS, tcp).");
		exit(1);
	}

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

**msg_clnt.c**

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

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

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

```
int *
printmessage_1(char **argp, CLIENT *clnt)
{
        static int clnt_res;

        memset((char *)&clnt_res, 0, sizeof(clnt_res));
        if (clnt_call (clnt, PRINTMESSAGE,
                (xdrproc_t) xdr_wrapstring, (caddr_t) argp,
                (xdrproc_t) xdr_int, (caddr_t) &clnt_res,
                TIMEOUT) != RPC_SUCCESS) {
                return (NULL);
        }
        return (&clnt_res);
}
```

**printproc.c**

```
#include <stdio.h>
#include "msg.h"                    /* msg.h generated by rpcgen */

int main(argc, argv)
        int argc;
        char *argv[];
{
        CLIENT *clnt;
        int *result;
        char *server;
        char *message;

        if (argc != 3) {
                fprintf(stderr, "usage : %s host message \n", argv[0]);
                exit(1);
        }

        server = argv[1];
        message = argv[2];
        /*
         * Create client "handle" used for
 * calling MESSAGEPROG on the server
         * designated on the command line.
         */
        clnt = clnt_create(server, MESSAGEPROG,
                                                PRINTMESSAGEVERS,
                                                "udp");
        if (clnt == (CLIENT *)NULL) {
                /*
                 * Couldn't establish connection
   * with server.
                 * Print error message and die.
                 */
```

```
                    clnt_pcreateerror(server);
                    exit(1);
          }
                    /*
           * Call the remote procedure
 * "printmessage" on the server
           */
          result = printmessage_1(&message, clnt);
          if (result == (int *)NULL) {
                    /*
                     * An error occurred while calling
   * the server.
                     * Print error message and die.
                     */
                    clnt_perror(clnt, server);
                    exit(1);
          }
          /* Okay, we successfully called
 * the remote procedure.
 */
          if (*result == 0) {
                    /*
                     * Server was unable to print
   * our message.
                     * Print error message and die.
                     */
                    fprintf(stderr,
                    "%s: could not print your message\n",argv[0]);
                    exit(1);
          }
          /* The message got printed on the
 * server's console
 */
          printf("Message delivered to %s\n",
                                        server);
          clnt_destroy( clnt );
          exit(0);
}
```

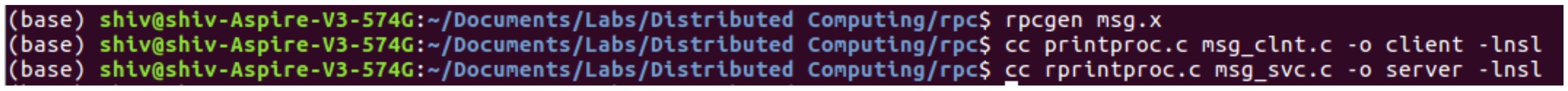
**rprintproc.c**

```
#include <stdio.h>
#include "msg.h"                              /* msg.h generated by rpcgen */

int *
printmessage_1_svc(msg, req)
          char **msg;
          struct svc_req *req;          /* details of call */
{
          static int result;                    /* must be static! */
          // FILE *f;
```

```
        // f = fopen("/dev/console", "w");
        // if (f == (FILE *)NULL) {
        //          result = 0;
        //          return (&result);
        // }
        // fprintf(f, "%s\n", *msg);
        // fclose(f);
    printf("%s\n",*msg);
        result = 1;
        return (&result);
}
```

**output:**



```
(base) shiv@shiv-Aspire-V3-574G:~/Documents/Labs/Distributed Computing/rpc$ rpcgen msg.x
(base) shiv@shiv-Aspire-V3-574G:~/Documents/Labs/Distributed Computing/rpc$ cc printproc.c msg_clnt.c -o client -lnsl
(base) shiv@shiv-Aspire-V3-574G:~/Documents/Labs/Distributed Computing/rpc$ cc rprintproc.c msg_svc.c -o server -lnsl
```