ASSIGNMENT 1

INTRODUCTION TO DISTRIBUTED SYSTEMS

UJJWAL SHARMA , BT18CSE021 , usharma@students.vnit.ac.in

Question 1 - Server Client Chat via R.P.C

Chat.x

Created a struct that will be used both as argument and return parameter so to convey messages

chat.x

chatRPC > code > chat.x

```
struct Message{
    char message[100];
};

program CHAT_PROG{
    version CHAT_VERS{
        Message chat(Message)=1;
    }=1;
}=0x1231231;
```

Then run rpcgen -aC chat.x to generate corresponding C files.
Then edit server and client files

chat_server.c

chat_server.c

chatRPC > code > chat_server.c

```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 "chat.h"

Message *
chat_1_svc(Message *argp, struct svc_req *rqstp)
{
    static Message  result;

    /*
     * insert server code here
     */
    printf("client ->> %s\n" , argp->message);
    printf("server ->> ");
    fgets(result.message, 100, stdin);
    // printf("%s\n", result.message);

    return &result;
}

```

chat_client.c

chat_client.c

chatRPC > code > chat_client.c

```c

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

    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    printf("press ctrl + c to exit\n");
    while(1){
        chat_prog_1 (host);
    }

exit (0);
}

```

Function : - chat_prog_1

```c
#endif  /* DEBUG */
    printf("client ->> ");
    fgets(chat_1_arg.message, 100, stdin);

    // printf("%s\n" , chat_1_arg.message);
    result_1 = chat_1(&chat_1_arg, clnt);
    if (result_1 == (Message *) NULL) {
        clnt_perror (clnt, "call failed");
    }else{
        printf("server >> %s\n", result_1->message);
    }
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}
```

Output Of Question 1

```
Terminal
$ ./chat_client localhost
press ctrl + c to exit
client ->> this is hi from client
```

```
Terminal
$ ./chat_server
```

```
Terminal
$ ./chat_client localhost
press ctrl + c to exit
client ->> this is hi from client
```

```
Terminal
$ ./chat_server
client ->> this is hi from client

server ->>
```

```
Terminal
$ ./chat_client localhost
press ctrl + c to exit
client ->> this is hi from client
server >> This is response from server

client ->>
```

```
Terminal
$ ./chat_server
client ->> this is hi from client

server ->> This is response from server
client ->> this is hi from client

server ->>
```

Question 2 - Factorial using R.P.C

Factorial.x

factorial.x

factorialRPC > factorial.x

```
struct number{
    int fact;
};

program FACT_PROG{
    version FACT_VERS{
        int factorial(number)=1;
    }=1;
}=0x1231232;
```

Factorial_server.c

factorial_server.c

factorialRPC > code > factorial_server.c

```c
int calculateFactorial(int fact){
    if(fact <0){
        return 0;
    }
    int result = 1;
    for(int i=1;i<=fact;i++){
        result*=i;
    }
    return result;
}

int *
factorial_1_svc(number *argp, struct svc_req *rqstp)
{
    static int  result;

    /*
     * insert server code here
     */
    printf("Calculating Factorial of %d ...\n",argp->fact);
    result = calculateFactorial(argp->fact);
    printf("Done\n");

    return &result;
}
```

factorial_client.c

Function - fact_prgm_1

factorial_client.c

factorialRPC > code > factorial_client.c

```c
        }
#endif  /* DEBUG */
    factorial_1_arg.fact = fact;
    result_1 = factorial_1(&factorial_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }else{
        printf("Factorial of %d is %d\n",fact , *result_1);
    }
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}


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

    if (argc != 3) {
        printf ("usage: %s server_host NUMBER\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    fact_prog_1 (host , atoi(argv[2]));
exit (0);
}

```

Output of answer 2

```
$ ./factorial_client localhost 5
Factorial of 5 is 120
$
```

```
$ ./factorial_server
Calculating Factorial of 5 ...
Done
```

```
$ ./factorial_client localhost 5
Factorial of 5 is 120
$ ./factorial_client localhost 10
Factorial of 10 is 3628800
$ ./factorial_client localhost 1
Factorial of 1 is 1
$
```

```
$ ./factorial_server
Calculating Factorial of 5 ...
Done
Calculating Factorial of 10 ...
Done
Calculating Factorial of 1 ...
Done
```

```
$ ./factorial_client localhost 5
Factorial of 5 is 120
$ ./factorial_client localhost 10
Factorial of 10 is 3628800
$
```

```
$ ./factorial_server
Calculating Factorial of 5 ...
Done
Calculating Factorial of 10 ...
Done
```

```
$ ./factorial_client localhost 5
Factorial of 5 is 120
$ ./factorial_client localhost 10
Factorial of 10 is 3628800
$ ./factorial_client localhost 1
Factorial of 1 is 1
$ ./factorial_client localhost 0
Factorial of 0 is 1
$
```

```
$ ./factorial_server
Calculating Factorial of 5 ...
Done
Calculating Factorial of 10 ...
Done
Calculating Factorial of 1 ...
Done
Calculating Factorial of 0 ...
Done
```

```
$ ./factorial_client localhost 5
Factorial of 5 is 120
$ ./factorial_client localhost 10
Factorial of 10 is 3628800
$ ./factorial_client localhost 1
Factorial of 1 is 1
$ ./factorial_client localhost 0
Factorial of 0 is 1
$ ./factorial_client localhost -10
Factorial of -10 is 0
$
```

```
$ ./factorial_server
Calculating Factorial of 5 ...
Done
Calculating Factorial of 10 ...
Done
Calculating Factorial of 1 ...
Done
Calculating Factorial of 0 ...
Done
Calculating Factorial of -10 ...
Done
```

Question 3 - Print date , time of the server using R.P.C

dateTime.x

dateTime.x

dateTimeRPC > dateTime.x

```
program TIME_PROG{
    version TIME_VERS{
        long dateTime(void)=1;
    }=1;

}= 0x1231233;
```

dateTime_server.c

dateTime_server.c

dateTimeRPC > code > dateTime_server.c

```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 "dateTime.h"
#include<time.h>

long *
datetime_1_svc(void *argp, struct svc_req *rqstp)
{
    static long  result;

    /*
     * insert server code here
     */
    printf("Sending current date and time . \n");
    result = time((long*)0);
    return &result;
}

```

dateTime_client.c

dateTime_client.c

dateTimeRPC > code > dateTime_client.c

```c
    }else{


    char dateTimeBuffer[100];
    strftime (dateTimeBuffer, 100, "%d-%m-%M %H:%M:%S", localtime (result_1));

    printf("Time on host : %s is :\n\t%s , ( %s )\n", host , dateTimeBuffer , ctime(result_1));
    }
#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];
    time_prog_1 (host);
exit (0);
}

```

Output of answer 3

```
Terminal
$ ./dateTime_client localhost

Terminal
$ ./dateTime_server
```

```
Terminal
$ ./dateTime_client localhost
Time on host : localhost is :
        17-02-23 11:23:32 , ( Thu Feb 17 11:23:32 2022
 )
$

Terminal
$ ./dateTime_server
Sending current date and time .
```

```
>_ Terminal
$ ./dateTime_client localhost
Time on host : localhost is :
        17-02-23 11:23:32 , ( Thu Feb 17 11:23:32 2022
 )
$ ./dateTime_client localhost
Time on host : localhost is :
        17-02-23 11:23:48 , ( Thu Feb 17 11:23:48 2022
 )
$
```

```
>_ Terminal
$ ./dateTime_server
Sending current date and time .
Sending current date and time .
```

UJJWAL SHARMA
BT18CSE021
usharma@students.vnit.ac.in

********** END **********