LAB ASSIGNMENT – 6

DISTRIBUTED SYSTEMS

NAME : SOURABH PATEL.

ID NO. : U19CS082

Que : Simulate RPC (Create any one procedure on remote machine and call it from local machine)

List of programs for RPC

1. **Find out the factorial of given number.**

✓ fact.x

```
program FACT_PROG{
version FACT_VERS{
        int FACT(int) = 1;
 }= 1;
}= 0x23451111;
```

✓ fact_client

```
/*
 * 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 "fact.h"
```

```c
void
fact_prog_1(char *host, int input)
{
    CLIENT *clnt;
    int  *result_1;
    int  fact_1_arg;

#ifndef DEBUG
    clnt = clnt_create (host, FACT_PROG, FACT_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif  /* DEBUG */

    fact_1_arg = input;

    result_1 = fact_1(&fact_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        int result=*result_1;
        if(result==-1)
            printf("\nInvalid input.\n");

        else
            printf("\nFactorial : %d\n",result);
    }
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}

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

    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }

    printf("Enter the number: ");
    scanf("%d",&input);

    host = argv[1];
```

```
    fact_prog_1 (host,input);
    exit (0);
}
```

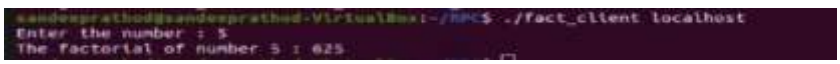
✓ fact_server

```
/*
 * 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 "fact.h"

int *
fact_1_svc(int *argp, struct svc_req *rqstp)
{
    static int  result;

    int temp = argp[0];

    printf("factorial(%d) called\n",temp);

    if(temp>=0){
        if(temp==0)
            temp=1;
        result = temp;

        while(temp!=1){
            temp--;
            result*=temp;
        }
    }
    else{
        result=-1;
    }

    return &result;
}
```

Output : ❼

Server:

Client :



2. Implement Calculator (Basic operation).

- ✓ calc.x

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

program CALC_PROG{
version CALC_VERS{
float add(numbers)=1;
float sub(numbers)=2;
float mul(numbers)=3;
float div(numbers)=4;
 }=1;
}= 0x123456;
```

- ✓ calc_client.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 "calc.h"


void
calc_prog_1(char *host,float x, float y, int choice)
{
    CLIENT *clnt;
    numbers  input_arg;
    float  *result;

#ifndef DEBUG
    clnt = clnt_create (host, CALC_PROG, CALC_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif  /* DEBUG */
```

```
    input_arg.a=x;
    input_arg.b=y;

    if(choice==1)
        result = add_1(&input_arg, clnt);
    if(choice==2)
            result = sub_1(&input_arg, clnt);
    if(choice==3)
            result = mul_1(&input_arg, clnt);
    if(choice==4)
            result = div_1(&input_arg, clnt);

    if (result == (float *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
            printf("\nResult = %f\n", *result);
    }

#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}


int
main (int argc, char *argv[])
{
    char *host;
    float x,y;
    int choice;

    if (argc < 2) {
        printf ("usage: %s server_host\n", argv[0]);
        exit (1);
    }
    host = argv[1];
    while(1){
        printf("\n1. Addition\n");
        printf("2. Substraction\n");
        printf("3. Multiplication\n");
        printf("4. Division\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);

        if(choice<1 || choice>5){
```

```
            printf("\nInvalid option.\n");
            continue;
        }

        if(choice == 5){
            printf("\nThank you.\n");
            break;
        }

        printf("Enter first number: ");
        scanf("%f",&x);
        printf("Enter second number: ");
        scanf("%f",&y);
        calc_prog_1 (host,x,y,choice);
    }

exit (0);
}
```

- ✓ calc_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 "calc.h"

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

float *
sub_1_svc(numbers *argp, struct svc_req *rqstp)
{
    static float  result;
```

```
    printf("substract(%f,%f) is called....\n",argp->a,argp->b );
    result = argp->a - argp->b;
    return &result;
}

float *
mul_1_svc(numbers *argp, struct svc_req *rqstp)
{
    static float  result;
    printf("multiply(%f,%f) is called....\n",argp->a,argp->b );
    result = argp->a * argp->b;
    return &result;
}

float *
div_1_svc(numbers *argp, struct svc_req *rqstp)
{
    static float  result;
    printf("division(%f,%f) is called....\n",argp->a,argp->b );
    result = argp->a / argp->b;
    return &result;
}
```

✓ Output : ❼ ✓

Server :

✓ Client :

3. Find out whether given number is Prime Number or not.

✓ Prime_check.x

```
program PRIMECHECK_PROG{
version PRIMECHECK_VERS{
int prime_check(int)=1;
 }=1;
}= 0x123456;
```

✓ Prime_check_client.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 "prime_check.h"


void
primecheck_prog_1(char *host, int x)
{
    CLIENT *clnt;
    int  *result_1;
    int  prime_check_1_arg;

#ifndef DEBUG
    clnt = clnt_create (host, PRIMECHECK_PROG, PRIMECHECK_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
```

```c
#endif  /* DEBUG */

    prime_check_1_arg = x;

    result_1 = prime_check_1(&prime_check_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        if(*result_1 == 0)
            printf("\n%d is not a prime number.\n",x);
        else
            printf("\n%d is a prime number.\n",x);
    }

#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}


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

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

    printf("\nEnter the number: ");
    scanf("%d",&x);

    primecheck_prog_1 (host,x);
exit (0);
}
```

✓ Prime_check_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 "prime_check.h"

int *
prime_check_1_svc(int *argp, struct svc_req *rqstp)
{
    static int  result;
    int i;

    printf("prime_check(%d) is called.....\n",*argp);

    if(*argp <= 1)
        result=0;
    else{
        result=1;
        for(i=2; i*i <= *argp; i++){
            if(*argp % i == 0){
                result=0;
                break;
            }
        }
    }
    return &result;
}
```

✓ Output : ❼ ✓

Server :

✓ Client :

4. Print out the Fibonacci series till the given number.

✓ fibb.x

```
struct data{
int arr[50];
int sz;
};

program FIBB_PROG{
version FIBB_VERS{
data find_fibb(int)=1;
 }=1;
}=0x111111;
```

✓ fibb_client.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 "fibb.h"


void
fibb_prog_1(char *host, int x)
{
    CLIENT *clnt;
    data  *result_1;
    int  find_fibb_1_arg;

#ifndef DEBUG
```

```
    clnt = clnt_create (host, FIBB_PROG, FIBB_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif  /* DEBUG */

    find_fibb_1_arg = x;

    result_1 = find_fibb_1(&find_fibb_1_arg, clnt);
    if (result_1 == (data *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        if(result_1->sz==0)
            printf("\nNo Fibonacci number.");
        else{
            printf("\nFibonacci numbers are : ");
            int i=0;
            for(i=0; i < result_1->sz; i++){
                printf("%d ",result_1->arr[i]);
            }
        }
        printf("\n");
    }

#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}


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

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

    printf("Enter the number: ");
    scanf("%d",&x);

    fibb_prog_1 (host,x);
```

```
exit (0);
}
```

- ✓ fibb_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 "fibb.h"

data *
find_fibb_1_svc(int *argp, struct svc_req *rqstp)
{
    static data  result;

    printf("fibb(%d) is called.....\n",*argp);

    int a=0,b=1;
    result.sz=0;
    while(b <= *argp){
        result.arr[result.sz]=b;
        result.sz++;
        b+=a;
        a=b-a;
    }
    return &result;
}
```

- ✓ Output : ❼ ✓

  Server :

- ✓ Client :

5. Find the maximum value of an array of integers using RPC.

find_max.x

```
struct data{
int arr[100];
int sz;
};

program FIND_MAX_PROG{
version FIND_MAX_VERS{
int find_max(data)=1;
 }=1;
}=0x1212111;
```

find_client

```
     /*
 * 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 "find_max.h"


void
find_max_prog_1(char *host,data *input)
{
    CLIENT *clnt;
    int  *result_1;
    data  find_max_1_arg;

#ifndef DEBUG
    clnt = clnt_create (host, FIND_MAX_PROG, FIND_MAX_VERS, "udp");
```

```
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }
#endif  /* DEBUG */

    find_max_1_arg = *input;
    result_1 = find_max_1(&find_max_1_arg, clnt);
    if (result_1 == (int *) NULL) {
        clnt_perror (clnt, "call failed");
    }
    else{
        printf("Maximum element : %d",*result_1);
    }
    printf("\n");
#ifndef DEBUG
    clnt_destroy (clnt);
#endif   /* DEBUG */
}


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

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

    printf("Enter the size of array: ");
    scanf("%d",&size);
    if(size>0){
        data input;
        input.sz=size;
        printf("Enter the elements: ");
        for(i=0; i<size; i++){
            scanf("%d",&(input.arr[i]));
        }
        find_max_prog_1 (host,&input);
    }
exit (0);
}
```

find_server

```
    /*
 * 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 "find_max.h"

int *
find_max_1_svc(data *argp, struct svc_req *rqstp)
{
    static int  result;
    int i;

    result=argp->arr[0];

    for(i=1; i < argp->sz; i++){
        if(result < argp->arr[i])
            result = argp->arr[i];
    }

    return &result;
}
```

Output : ❼ Server :

Client :