Практичне заняття

**Виклик віддалених процедур (RPC)**

Мета: знасвоєння навичок створення програм з використанням RPC для обміну даними між віддаленими машинами.

Виклик віддалених процедур (Remote procedure calls, RPC) це механізм, за допомогою якого один процес активізує інший процес на цій самій або віддаленій машині для виконання якоїсь функції від свого імені. RPC нагадує виклик локальної функції: процес викликає функцію і передає їй дані, а потім очікує, коли вона поверне результат. Специфіка полягає в тому, що цю функцію виконує інший процес. Така взаємодія процесів обов'язково перебігає за схемою клієнт/сервер, у якій процес, що активізує RPC, є клієнтським, а процес, що виконує RPC-функцію, — серверним. Серверний процес забезпечує доступ до однієї або декількох сервісних функцій, які можуть викликатися його клієнтами. Для розробки розподілених програм достатньо знати тільки інтерфейси (типи аргументів і зворотні значення).

Приклад програми, яка використовує RPC для обміну повідомленнями між клієнтом та сервером:

Приклад опису інтерфейсів (назва файлу — add.x):

struct numbers{

int a;

int b;

};

program ADD\_PROG{

version ADD\_VERS {

int add(numbers)=1;

}=1;

}=0x23451111;

Для компіляції використано rpcgen, який генерує заголовочний файл (add.h) файли клієнта (add\_clnt.c) і сервера (add\_svc.c) на мові С:

[tb015@plus7 test]$ rpcgen add.x

[tb015@plus7 test]$ ls

add\_clnt.c add.h add\_svc.c add.x add\_xdr.c

Приклад тіла процедури (назва файлу — 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;

}

Приклад клієнта (назва файлу — 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 \*/

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);

}

#ifndef DEBUG

clnt\_destroy (clnt);

#endif /\* DEBUG \*/

}

int

main (int argc, char \*argv[])

{

char \*host;

if (argc < 4) {

printf ("usage: %s server\_host NUMBER NUMBER\n", argv[0]);

exit (1);

}

host = argv[1];

add\_prog\_1 (host, atoi(argv[2]), atoi(argv[3]));

exit (0);

}

Приклад компіляції клієнта:

[tb015@plus7 example]$ gcc ./add\_client.c add\_clnt.c -o add\_client

і сервера:

[tb015@plus7 example]$ gcc ./add\_server.c add\_svc.c -o add\_server

Приклад роботи:

Клієнт:

[tb015@plus7 example]$ ./add\_client localhost 5 3

Result:8

Сервер:

[tb015@plus7 example]$ ./add\_server

add (5, 3) is called

## Завдання

1. Скомпілюйте програму прикладу та перевірте її працездатність.
2. Розберіть, що ця програма робить і як реалізована. Зверніть увагу, що всі аргументи та результати функцій передаються через вказівники.
3. Аналогічно до прикладу вище реалізувати програму, яка передає за допомогою клієнта на сервер текстове повідомлення, а сервер повертає на клієнт текстову відповідь.

**Контрольні питання**

1. Що робить rpcgen?
2. Які переваги та недоліки має застосування RPC?

**Література**

1. Chan T. Unix System Programming Using C++. Upper Saddle River N.J: Prentice Hall PTR, 1997. 598 p.
2. rpcgen Programming Guide. [Електронний ресурс]: <https://docs.oracle.com/cd/E19504-01/802-5885/6i9k4u0ap/index.html> (дата звернення: 27.11.2023)