

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»**

Факультет безопасности информационных технологий

Дисциплина:
«Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

Выполнила:

Нгуен Хонг Хань N3249

(подпись)

Проверил:

Савков Сергей Витальевич

(подпись)

Санкт-Петербург

2022г.

Задание

Одно из

1. Написать фильтр сетевых пакетов на основе nfqueue и iptables и протестировать скорость работы
2. Протестировать работу сокетов tcp при различных настройках setsockopt

Сложный вариант (одно из)

1. Написать фильтр пакетов на основе интерфейса netfilter
2. Реализовать rpc-программу для linux с поддержкой аутентификации (rpcinfo, rpcbind)

1. Протестировать работу сокетов tcp при различных настройках setsockopt

1.1. SO_REUSEADDR

SO_REUSEADDR позволяет запустить сервер прослушивателя и раскладывающую известные порты, даже если ранее установленный порт используется в качестве подключения их локальных портов, все еще существует. Если мы не настроим SO_REUSEADDR перед вызовом Bind, то второй процесс будет ошибкой при вызове функции BIND (адрес уже используется).

```
hanh@ubuntu:~$ gcc server.c -o s1
hanh@ubuntu:~$ gcc server.c -o s2
hanh@ubuntu:~$ ./s1
server bind success, 0.0.0.0:9527
```

```
hanh@ubuntu: ~
```

```
hanh@ubuntu:~$ ./s2
bind fail: Address already in use
hanh@ubuntu:~$
```

```
int flag = 1;
if (-1 == setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, & flag, sizeof(flag))) {
    Perror("setsockopt fail");
}
```

```
hanh@ubuntu:~$ gcc server.c -o s1
hanh@ubuntu:~$ gcc server.c -o s2
hanh@ubuntu:~$ ./s1
server bind success, 0.0.0.0:9527
connection from 127.0.0.1, port 45274
```

```
hanh@ubuntu: ~
```

```
hanh@ubuntu:~$ telnet 0.0.0.0 9527
Trying 0.0.0.0...
Connected to 0.0.0.0.
```

```
hanh@ubuntu: ~
hanh@ubuntu:~$ ./s2
server bind success, 0.0.0.0:9527
connection from 127.0.0.1, port 45278
```

```
hanh@ubuntu: ~
```

```
hanh@ubuntu:~$ telnet 0.0.0.0 9527
Trying 0.0.0.0...
Connected to 0.0.0.0.
```

1.2. SO_RCVBUF, SO_SNDBUF

SO_RCVBUF: Размер приемного буфера в байтах. Задать или получить максимальный размер буфера приёма сокета (в байтах). Ядро удваивает это значение (для пространства под учёт ресурсов (bookkeeping overhead)) при установке этого параметра с помощью `setsockopt(2)`, и это удвоенное значение возвращается `getsockopt(2)`.

SO_SNDBUF: Размер буфера передачи в байтах

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
void error_handling(char *message);

int main(int argc, char *argv[])
{
    int sock;
    int snd_buf = 1024 * 3, rcv_buf = 1024 * 3;
    int state;
    socklen_t len;

    sock = socket(PF_INET, SOCK_STREAM, 0);
    state = setsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void *)&rcv_buf, sizeof(rcv_buf));
    if (state)
        error_handling("setsockopt() error!");

    state = setsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void *)&snd_buf, sizeof(snd_buf));
    if (state)
        error_handling("setsockopt() error!");

    len = sizeof(snd_buf);
    state = getsockopt(sock, SOL_SOCKET, SO_SNDBUF, (void *)&snd_buf, &len);
    if (state)
        error_handling("getsockopt() error!");

    len = sizeof(rcv_buf);
    state = getsockopt(sock, SOL_SOCKET, SO_RCVBUF, (void *)&rcv_buf, &len);
    if (state)
        error_handling("getsockopt() error!");

    printf("Input buffer size: %d \n", rcv_buf);
    printf("Output buffer size: %d \n", snd_buf);
    return 0;
}

void error_handling(char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```

```
hanh@ubuntu:~$ ./setbuf
Input buffer size: 6144
Output buffer size: 6144
hanh@ubuntu:~$
```

2. Реализовать грс-программу для linux с поддержкой аутентификации (rpcinfo, rpcbind)

```
hanh@ubuntu:~$ sudo apt-get install rpcbind
```

```
hanh@ubuntu:~$ rpcinfo
program version netid address service owner
100000 4 tcp6 ::0.111 portmapper superuser
100000 3 tcp6 ::0.111 portmapper superuser
100000 4 udp6 ::0.111 portmapper superuser
100000 3 udp6 ::0.111 portmapper superuser
100000 4 tcp 0.0.0.0.0.111 portmapper superuser
100000 3 tcp 0.0.0.0.0.111 portmapper superuser
100000 2 tcp 0.0.0.0.0.111 portmapper superuser
100000 4 udp 0.0.0.0.0.111 portmapper superuser
100000 3 udp 0.0.0.0.0.111 portmapper superuser
100000 2 udp 0.0.0.0.0.111 portmapper superuser
100000 4 local /run/rpcbind.sock portmapper superuser
100000 3 local /run/rpcbind.sock portmapper superuser
```

**Создадим IDL файл. Он будет использоваться для генерации шаблона
rpc программы. (file IDL.x)**

```
/*The IDL File --- name IDL.x*/

/*Structure to hold the 2 values to be used in computation*/

struct values{
float num1;
float num2;
char operation;
};

/*Programme, version and procedure definition*/

program COMPUTE{
version COMPUTE_VERS{
float ADD(values) =1;
float SUB(values)=2;
float MUL(values)=3;
float DIV(values)=4;
} = 6;

} = 456123789;
```

Сгенерируем нужные нам файлы сервера и клиента с помощью команды

```
hanh@ubuntu:~/lab9$ rpcgen -a -C IDL.x
hanh@ubuntu:~/lab9$ ls
IDL_client.c IDL_clnt.c IDL.h IDL_server.c IDL_svc.c IDL.x IDL_xdr.c Makefile.IDL
```

IDL_server.c

```
#include "IDL.h"
#include <stdio.h>

float *
add_6_svc(values *argp, struct svc_req *rqstp)
{
    static float result;
    result = argp->num1 + argp->num2;
    return &result;
}

float *
sub_6_svc(values *argp, struct svc_req *rqstp)
{
    static float result;
    result = argp->num1 - argp->num2;
    return &result;
}

float *
mul_6_svc(values *argp, struct svc_req *rqstp)
{
    static float result;
```

```

        result = argp->num1 * argp->num2;
        return &result;
    }

    float *
    div_6_svc(values *argp, struct svc_req *rqstp)
    {
        static float result;
        result = argp->num1 / argp->num2;
        return &result;
    }

```

IDL_client.c

```

#include "IDL.h"
#include <stdio.h>

float compute_6(char *host, float a, float b, char op)
{
    CLIENT *clnt;
    float *result_1;
    values add_6_arg;
    float *result_2;
    values sub_6_arg;
    float *result_3;
    values mul_6_arg;
    float *result_4;
    values div_6_arg;

    if(op=='+'){

        add_6_arg.num1=a;
        add_6_arg.num2=b;
        add_6_arg.operation=op;

        clnt = clnt_create (host, COMPUTE, COMPUTE_VERS, "udp");
        if (clnt == NULL) {
            clnt_pcreateerror (host);
            exit (1);
        }

        result_1 = add_6(&add_6_arg, clnt);
        if (result_1 == (float *) NULL) {
            clnt_perror (clnt, "call failed");
        }

        clnt_destroy (clnt);

        return (*result_1);
    }

    else if(op=='-'){
        sub_6_arg.num1=a;
        sub_6_arg.num2=b;
        sub_6_arg.operation=op;

        clnt = clnt_create (host, COMPUTE, COMPUTE_VERS, "udp");
        if (clnt == NULL) {
            clnt_pcreateerror (host);
            exit (1);
        }

        result_2 = sub_6(&sub_6_arg, clnt);
        if (result_2 == (float *) NULL) {

```

```

        clnt_perror (clnt, "call failed");
    }

    clnt_destroy (clnt);

    return (*result_2);
}

else if(op=='*'){
    mul_6_arg.num1=a;
    mul_6_arg.num2=b;
    mul_6_arg.operation=op;

    clnt = clnt_create (host, COMPUTE, COMPUTE_VERS, "udp");
    if (clnt == NULL) {
        clnt_pcreateerror (host);
        exit (1);
    }

    result_3 = mul_6(&mul_6_arg, clnt);
    if (result_3 == (float *) NULL) {
        clnt_perror (clnt, "call failed");
    }

    clnt_destroy (clnt);

    return (*result_3);
}

else if(op=='/'){
    if(b==0){
        printf("You are trying to divide by zero. Please insert a valid
number.\n");
        exit(0);
    }
    else{
        div_6_arg.num1=a;
        div_6_arg.num2=b;
        div_6_arg.operation=op;

        clnt = clnt_create (host, COMPUTE, COMPUTE_VERS, "udp");
        if (clnt == NULL) {
            clnt_pcreateerror (host);
            exit (1);
        }

        result_4 = div_6(&div_6_arg, clnt);
        if (result_4 == (float *) NULL) {
            clnt_perror (clnt, "call failed");
        }

        clnt_destroy (clnt);

        return (*result_4);
    }
}

}

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

```

```

float number1,number2;
char oper;
printf("Enter the 2 numbers followed by the operation to perform:\n");
scanf("%f",&number1);
scanf("%f",&number2);
scanf("%s",&oper);

host = argv[1];
printf("Answer= %f\n",compute_6 (host,number1,number2,oper));
exit(0);

}

```

Соберем

```

hanh@ubuntu:~/lab9$ make -f Makefile.IDL
cc -g -c -o IDL_clnt.o IDL_clnt.c
cc -g -c -o IDL_client.o IDL_client.c
cc -g -c -o IDL_xdr.o IDL_xdr.c
cc -g -o IDL_client IDL_clnt.o IDL_client.o IDL_xdr.o -lnsl
cc -g -c -o IDL_svc.o IDL_svc.c
cc -g -c -o IDL_server.o IDL_server.c
cc -g -o IDL_server IDL_svc.o IDL_server.o IDL_xdr.o -lnsl
hanh@ubuntu:~/lab9$ ls
IDL_client  IDL_clnt.c  IDL_server  IDL_svc.c  IDL_xdr.c
IDL_client.c  IDL_clnt.o  IDL_server.c  IDL_svc.o  IDL_xdr.o
IDL_client.o  IDL.h      IDL_server.o  IDL.x      Makefile.IDL

```

Запустим сервер и клиента в разных окнах терминала

```

hanh@ubuntu:~/lab9$ sudo ./IDL_server
[sudo] password for hanh:

```

```

hanh@ubuntu:~/lab9$ sudo ./IDL_client localhost
[sudo] password for hanh:
Enter the 2 numbers followed by the operation to perform:
1 2 *
Answer= 2.000000

```

Авторизация прошла успешно, и программа дала верный ответ

Попробуем зайти под другим логином и ввести неверный пароль

```

hanh@ubuntu:~$ sudo ./IDL_client localhost
[sudo] password for hanh:
Sorry, try again.

```

Выводы: В ходе лабораторной работы мы разобрались с утилитой грсген, работой с RPC, а также могли написать простое грс-приложение.