RTFM: Linux, DevOps и системное администрирование
DevOps-инжиниринг и системное администрирование. Случаи из практики.

C: сокеты и пример модели client-server

Автор: setevoy | 05/16/2017 1 Comment

Rate this (175 Votes)

Перевод с дополнениями. Оригинал – тут>>>.

Как правило – два процесса общаются друг с другом с помощью одного из *Inter Process Communication* (IPC) механизма ядра, таких как:

- *pipe*
- *очереди сообщений (Message queues)*
- *общая память (shared memory)*

Кроме перечисленных **IPC** – в ядре присутствует много других возможностей, но что если процессам необходимо обмениваться данными по сети?

Тут используется ещё один механизм **IPC** – сокеты.

Содержание

Что такое сокет?

> *Сокеты (англ. socket – разъём) – название программного интерфейса для обеспечения обмена данными между процессами. Процессы при таком обмене могут исполняться как на одной ЭВМ, так и на различных ЭВМ, связанных между собой сетью. Сокет – абстрактный объект, представляющий конечную точку соединения.*

Кратко говоря – существует два типа сокетов – **UNIX**-сокеты (или сокеты домена **UNIX** – *Unix domain sockets*) и **INET**-сокеты (*IP-сокеты, network sockets*).

UNIX-сокеты чвляются частью механизма **IPC** и позволяют обмен данными в обоих направлениях между процессами, работающими на одной машине.

INET-сокеты в свою очередь представляют собой механизм, позволяющий выполнять коммуникацию между процессами по сети.

Грубо говоря – если **UNIX**-сокет использует файл в файловой системе, то **INET**-сокет – требует присваивания сетевого адреса и порта.

Больше про сокеты:

- Unix domain socket
- Network socket
- Что такое сокет?
- Сокеты
- What is a socket?

Коммуникация в среде **TCP/IP** происходит по клиент-серверной модели, т.е. – клиент инициализирует связь, а сервер его принимает.

Ниже – пример сервера, который будет работать как демон и ожидать подключения клиента, а при инициализации клиентом соединения – передаст ему дату и время.

Socket сервер

Наш сервер будет выглядеть следующим образом:

```
01 #include <sys/socket.h>
```

```c
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>

int main(int argc, char *argv[]) {
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    char sendBuff[1025];
    time_t ticks;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);

    bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

    listen(listenfd, 10);

    while(1) {
        connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);

        ticks = time(NULL);
        snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n", ctime(&ticks))
        write(connfd, sendBuff, strlen(sendBuff));

        close(connfd);
        sleep(1);
     }
}
```

Собираем, запускаем:

Terminal

```
$ gcc server.c -o server
$ ./server &
[1] 26182
```

Re-play Copy to Clipboard Pause Full View

Проверяем:

Terminal

```
$ netstat -anp --tcp | grep server
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
tcp        0      0 0.0.0.0:5000            0.0.0.0:*
```

Re-play Copy to Clipboard Pause Full View

Флаг --tcp для netstat указывает на то, что требуется вывести информацию только по **INET**-сокетам.

Самый простой способ получить данные от нашего сервера – с помощью telnet, проверяем ещё раз:

Terminal

```
$ telnet localhost 5000
Trying ::1...
Connection failed: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Tue May 16 12:43:24 2017
Connection closed by foreign host.
```

Re-play Copy to Clipboard Pause Full View

Данные получены:

```
…
Tue May 16 12:43:24 2017
…
```

Теперь – давайте рассмотрим сам код сервера.

```
...
listenfd = socket(AF_INET, SOCK_STREAM, 0);
...
```

- с помощью вызова функции socket() в области ядра создаётся неименованный сокет, и возвращается его *socket descriptor*
- первым аргументом этой функции передаётся тип домена. Т.к. мы будем использовать сеть – то используем тип сокета AF_INET (IPv4).
- вторым аргументом – SOCK_STREAM, который указывает на тип протокола. Для **TCP** – это будет SOCK_STREAM, для **UDP** – SOCK_DGRAM
- третий аргумент оставляем по умолчанию – тут ядро само решит какой тип протокола использовать (т.к. мы указали SOCK_STREAM – то будет выбран **TCP**)

Далее – вызывается функция bind():

```
...
bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
...
```

- bind() создаёт сокет используя параметры из структуры serv_addr (протокол, IP-адрес и порт)

```
...
listen(listenfd, 10);
...
```

- вызов функции listen() со вторым аргументом 10 указывает на макс. допустимое кол-во подключений. Первым аргументом – передаётся дескриптор сокета, который необходимо прослушивать.

Далее – accept():

```
...
connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);
...
```

- сервер запускает бесконченый цикл, ожидая входящего соединения, и вызывает accept(), как только соединение установлено. В свою очередь accept() создаёт новый сокет для каждого соединения, вовзращает дескриптор сокета
- как только соединение установлено (т.е. сокет создан) – функция snprintf() вписывает время и дату в буфер, после чего вызывается write(), которая вписывает данные из буфера в сокет

Socket клиент

Перейдём ко второй программе – клиенту.

Код её будет выглядеть следующим образом:

```
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <arpa/inet.h>

int main(int argc, char *argv[]) {
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;

    if(argc != 2) {
        printf("\n Usage: %s <ip of server> \n",argv[0]);
        return 1;
    }

    memset(recvBuff, '0',sizeof(recvBuff));
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket \n");
        return 1;
    }

    memset(&serv_addr, '0', sizeof(serv_addr));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000);

    if(inet_pton(AF_INET, argv[1], &serv_addr.sin_addr)<=0)
    {
        printf("\n inet_pton error occured\n");
        return 1;
```

```
    }

    if( connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
    {
       printf("\n Error : Connect Failed \n");
       return 1;
    }

    while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
    {
        recvBuff[n] = 0;
        if(fputs(recvBuff, stdout) == EOF)
        {
            printf("\n Error : Fputs error\n");
        }
    }

    if(n < 0)
    {
        printf("\n Read error \n");
    }

    return 0;
}
```

Кратко рассмотрим его:

```
...
if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
...
```

- аналогично серверу – создаём сокет

```
...
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(5000);

    if(inet_pton(AF_INET, argv[1], &serv_addr.sin_addr)<=0)
...
```

- в структуру sockaddr_in с именем serv_addr заносятся протокол, порт (*5000*) и адрес сервера (первый аргумент – argv[1])

```
...
if( connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
...
```

- функция connect() пытается установить соединение с хостом, используя данные из структуры serv_addr

```
...
    while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
...
```

И в конце-концов – клиент с помощью read() получает данные из своего сокета, в который поступают данные от сокета на сервере.

Собираем клиент, и пробуем подключиться к нашему серверу:

Terminal

```
$ gcc client.c -o client
$ ./client
Usage: ./client <ip of server>
$ ./client 127.0.0.1
Tue May 16 12:13:54 2017
```

Re-play Copy to Clipboard Pause Full View

Готово.

Ссылки по теме

Сокеты

Сокеты

Berkeley sockets

Interprocess Communication Mechanisms

Что такое сокет?

Beej’s Guide to Network Programming Using Internet Sockets

Similar posts

- 03/10/2018 What is: Linux namespaces, примеры PID и Network namespaces (0)
- 09/17/2017 Linux: C – адресное пространство процесса (0)
- 08/08/2017 C: “мониторинг” NGINX с помощью AF_INET (0)
- 07/31/2017 C: создание и применение shared library в Linux (0)
- 02/17/2016 Linux: strace – отслеживаем выполнение процесса (0)

Раздел: C/C++ GNU/Linux utils HOWTO's Networking Operating systems Scripting/coding UNIX/Linux Web-services Метки: C++, Linux, socket