Мрежово програмиране

Програмиране на приложно ниво – I част

доц. д-р Йордан Денев
denev@fmi.uni-sofia.bg

- В този раздел ще разгледаме принципите на приложното прорграмиране.
- Материалът ще бъде илюстриран основно с примери на C в среда на UNIX.
- Бегло ще се разгледат примери на Java, както и приложното мрежово програмиране в среда на MS Windows.

Взаимодействие между клиент и сървер

Адресиране на socket

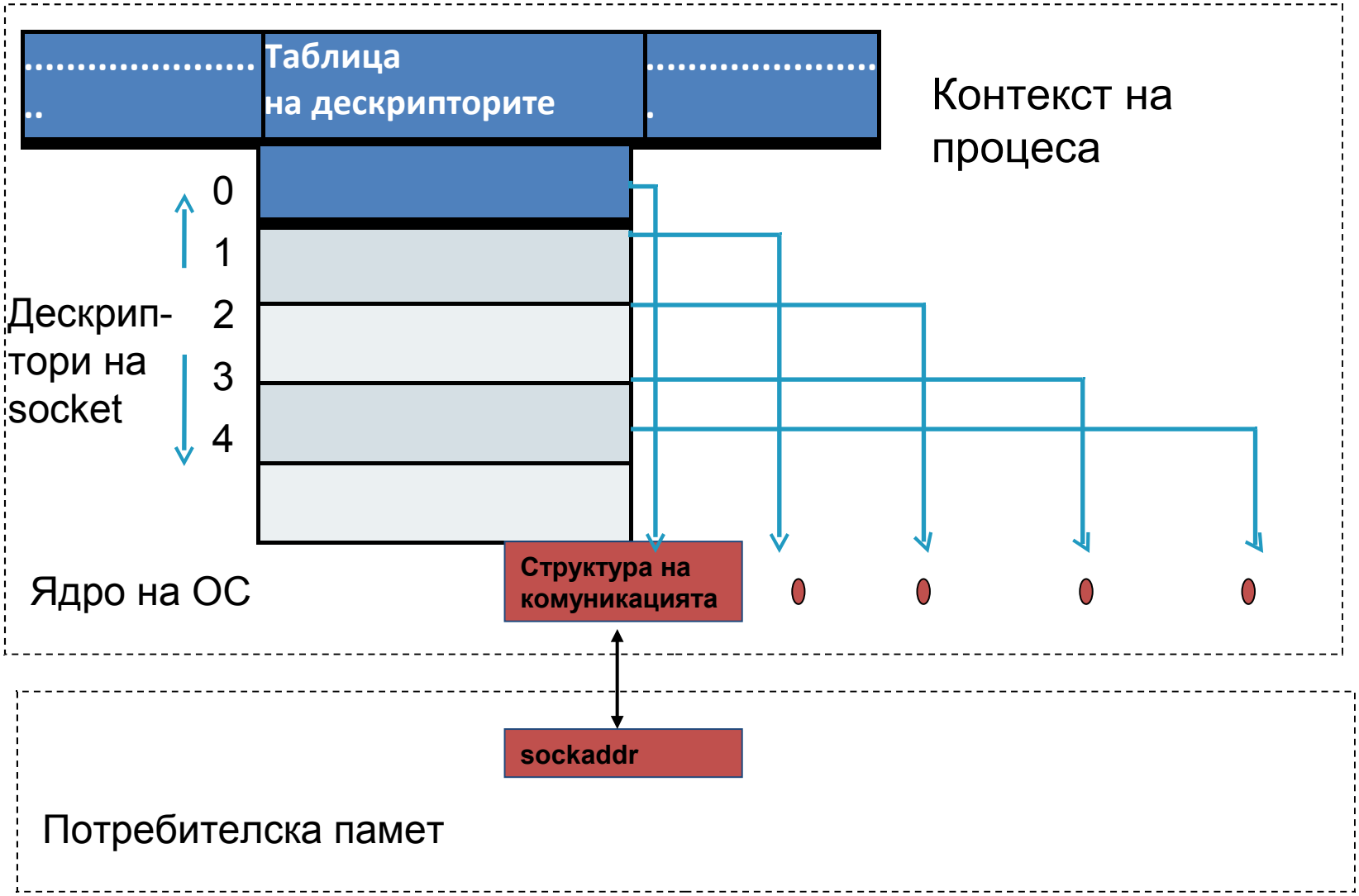
- Следните структури наменоват socket:

```
struct sockaddr_in {
   short int sin_family;          /*Protocol family*/
   unsigned short int sin_port; /*Port number*/
   struct in_addr sin_addr;     /*Internet address*/
};

sin_family = AF_INET    /*Protocol type*/

struct in_addr{
   unsigned long int     s_addr;
}
```

Управляващи таблици



Инициализация на socket

- Създаване на socket на услугата

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int domain, int type, int
  protocol);
```

domain –> AF_UNIX, AF_INET,AF_ISO

type -> SOCK_STREAM, SOCK_DGRAM

protocol -> 0

Return -> sockfd

- Н аименование на услугата

```
#include <sys/types.h>
#include <sys/socket.h>
int bind(int sockfd, struct sockaddr
  *my_addr, int addrlen);
```

sockfd -> дескриптор на socket-а (стойността върната от **socket**).

sockaddr -> указател към структура, наименоваща socket

addrlen -> дължината на горната структура

Return -> 0 (0.K.), -1 (error)

- Дефиниране на опашка

```
int listen (int sockfd, int backlog);
```

sockfd -> стойността върната от **socket**

backlog -> дължината на опашката със заявки, чакащи за сървера

Return -> 0 (O.K.), -1 (error)

- Създаване на socket на връзката

```
#include <sys/types.h>
#include <sys/socket.h>
int accept(int sockfd, struct sockaddr *addr, socklen_t
   *addrlen);
```

sockfd -> дескриптор на **socket-а** на услугата.

addr -> адрес на структура, в която **accept** записва данните на клиента, който се обръща за услугата.

addrlen ->указател към цяла стойност, задаваща дължината на горната структура.

Return-> дескриптор на socket-а на връзката.

❑ Четене/писане в комуникационния канал

```
#include <unistd.h>
ssize_t read(int sockfd, void *buf, size_t count);
ssize_t  write(int sockfd, void *buf, size_t count);
```

❑ Затваряне на канала

```
#include <unistd.h>
int close(int sockfd)
```

Прост сървер

```
/*  Make the necessary includes and set up the variables.
   */
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

int main()
{
    int server_sockfd, client_sockfd;
    int server_len, client_len;
    struct sockaddr_in server_address;
    struct sockaddr_in client_address;
```

```
/*  Create an unnamed socket for the server.  */

    server_sockfd = socket(AF_INET, SOCK_STREAM, 0);

/*  Name the socket.  */

    server_address.sin_family = AF_INET;
 /* Point interface (INADDR_ANY  if any) */
    server_address.sin_addr.s_addr = inet_addr("127.0.0.1");
    server_address.sin_port = 9734;
    server_len = sizeof(server_address);
    bind(server_sockfd, (struct sockaddr *)&server_address,
  server_len);
```

```
/*  Create a connection queue and wait for clients.
   */
    listen(server_sockfd, 5);
    while(1) {
        char ch;
        printf("server waiting\n");


/*  Accept a connection.  */
        client_len = sizeof(client_address);
        client_sockfd = accept(server_sockfd,
            (struct sockaddr *)
        &client_address, &client_len);
```

```
/*  We can now read/write to client on
   client_sockfd.  */

        while( read(client_sockfd, &ch, 1) !=0)
              printf ("server receives =
                              %c\n",ch);
        printf("server closes\n");
        close(client_sockfd);
     }
}
```

Свързване на клиент към сървер

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


connect(client_sockfd, (struct sockaddr
  *)&client_address, client_len);
```

Прост клиент

```
/*  Make the necessary includes and set up the
   variables.  */
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
```

```
int main()
{
    int sockfd;
    int client_len;
    int i;
    struct sockaddr_in client_address;
    int result;
    char ch = 'A';


/*  Create a socket for the client.  */

    sockfd = socket(AF_INET, SOCK_STREAM, 0);
```

```
/*  Name the socket, as agreed with the server.  */

    client_address.sin_family = AF_INET;
    client_address.sin_addr.s_addr =
  inet_addr("127.0.0.1");
    client_address.sin_port = 9734;
    client_len = sizeof(client_address);
```

```
/*  Now connect our socket to the server's socket.
   */

    result = connect(sockfd, (struct sockaddr
   *)&client_address, client_len);

    if(result == -1) {
        perror("oops: clienta");
        exit(1);
    }
```

```
/*  We can now read/write via sockfd.  */

    for (i=0; i<=9;i++) {
        write(sockfd, &ch,1);
        printf("client sends= %c\n", ch);
        ch++;
    }
    close(sockfd);
    exit(0);
}
```

Проблеми на представянето

- Big-endian computers

- Little-endian computers

Стандартно представяне

```
#include <netinet/in.h>
unsigned long int htonl(unsigned long int host)
unsigned short int htons(unsigned short int host)
unsigned long int ntohl(unsigned long int net)
unsigned short int ntohs(unsignet short int net)
```

Пример:

```
server_address.sin_port = htons(9734);
```