

© 2002-2015 Volnys Bernal 1

## Cliente UDP

**Volnys Borges Bernal**  
 volnys@lsi.usp.br  
 Departamento de Sistemas Eletrônicos  
 Escola Politécnica da USP



© 2002-2015 Volnys Bernal 2

## Agenda

- **Resumo das chamadas sockets para UDP**
- **Uso das chamadas sockets para UDP**
  - ❖ Chamada socket()
  - ❖ Chamada connect()
  - ❖ Chamada send()
  - ❖ Chamada recv()
  - ❖ Chamada close()

© 2002-2015 Volnys Bernal 3

## Resumo das chamadas UDP



© 2002-2015 Volnys Bernal 4

## Resumo de Chamadas UDP

Lado Cliente

```

socket ()
connect ()
send ()
recv ()
close ()

```

Lado Servidor

```

socket ()
bind ()
recvfrom ()
sendto ()
close ()

```

*Pré define o parceiro de comunicação para todo send()*

© 2002-2015 Volnys Bernal 5

## Resumo de Chamadas UDP

Lado Cliente

```

socket ()
sendto ()
recvfrom ()
close ()

```

Lado Servidor

```

socket ()
bind ()
recvfrom ()
sendto ()
close ()

```

*Informa o parceiro de comunicação a cada chamada*

© 2002-2015 Volnys Bernal 6

## Chamada socket()



© 2002-2015 Volhys Bernal 7

## Chamada socket()

- ❑ **Objetivo**
  - ❖ Criar um novo socket (plug de comunicação). Aloca estruturas de dados no sistema operacional para suportar a comunicação.
- ❑ **Resultado**
  - ❖ Retorna o descritor de arquivo (número inteiro).
- ❑ **Sintaxe**

```
sd = socket (int domain, int type, int protocol)
```
- ❑ **Observação:**
  - ❖ Quando um socket é criado, não possui nenhuma informação sobre o parsocket (endereços IPs e portas dos parceiros).

© 2002-2015 Volhys Bernal 8

## Chamada socket()

- ❑ **Sintaxe geral**

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

**Socket descriptor**

**Para PF\_INET use 0**

- Se SOCK\_DGRAM - UDP
- Se SOCK\_STREAM - TCP

**Pilha de protocolos:**

- PF\_LOCAL (file)
- PF\_INET (IPv4)
- PF\_INET6 (IPv6)
- PF\_X25 (X25)

**Tipo da comunicação:**

- SOCK\_STREAM (TCP)
- SOCK\_DGRAM (UDP)
- SOCK\_RAW (IP)

© 2002-2015 Volhys Bernal 9

## Chamada socket()

- ❑ **Tipo de serviço**
  - ❖ SOCK\_STREAM
    - Para ser utilizado com o protocolo TCP
    - Canal de comunicação full duplex
    - Fluxo de bytes sem delimitação
    - Chamadas para transmissão e recepção de dados:
      - read(), write() ou send(), recv()
  - ❖ SOCK\_DGRAM
    - Para ser utilizado com o protocolo UDP
    - Datagrama (mensagens)
    - Chamadas para transmissão e recepção de dados:
      - send(), sendfrom(), recv() ou recvfrom()
  - ❖ SOCK\_RAW
    - Permite acesso a protocolos de mais baixo nível
    - Datagrama (mensagens)
    - Chamadas para transmissão e recepção de dados:
      - send(), recv()

© 2002-2015 Volhys Bernal 10

## Chamada socket()

- ❑ **Para criar um socket TCP**

```
#include <sys/socket.h>
sd = socket(AF_INET, SOCK_STREAM, 0);
```
- ❑ **Para criar um socket UDP**

```
#include <sys/socket.h>
sd = socket(AF_INET, SOCK_DGRAM, 0);
```

© 2002-2015 Volhys Bernal 11

## Chamada socket()

- ❑ **Exemplo de criação de socket UDP**

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

int sd; // socket descriptor
. . .
sd = socket(PF_INET, SOCK_DGRAM, 0);
if (sd == -1)
{
    perror("Erro na chamada socket");
    exit(1);
}
. . .
```

© 2002-2015 Volhys Bernal 12

## Chamada Connect()



© 2002-2015 Volhys Bernal 13

## Chamada connect()

- **Objetivo**
  - ✦ Estabelecer uma sessão de comunicação TCP, UDP ou IP
- **Detalhamento**
  - ✦ Deve ser utilizado somente no lado cliente
  - ✦ UDP:
    - Informa ao sistema operacional o socket address (IP+porta) do parceiro de comunicação
    - Não são enviados datagramas
  - ✦ TCP:
    - Informa ao sistema operacional o socket address (IP+porta) do parceiro de comunicação
    - Estabelece a conexão TCP (*3 way handshake*)

© 2002-2015 Volhys Bernal 14

## Chamada connect()

- **Sintaxe**

```
#include <netdb.h>
int connect(int sd,
            struct sockaddr *serversockaddr,
            int size)
```

© 2002-2015 Volhys Bernal 15

## Chamada connect()

```
#include <netdb.h>

int          status;           //estado da chamada
struct sockaddr_in serveraddr; //endereço do servidor
...

// define endereço destino
serveraddr.sin_family = AF_INET;
serveraddr.sin_port   = htons(serverport);
status = inet_pton(AF_INET, stringIP, &serveraddr.sin_addr);
if (status <= 0)
    perror("Erro na conversão do endereço IP");

// ativa connect
status = connect( sd,
                 (struct sockaddr *)&serveraddr,
                 sizeof(serveraddr) );

if (status != 0)
    perror("Erro na chamada connect");
```

© 2002-2015 Volhys Bernal 16

## Chamada connect

- **Exemplo de iniciação da estrutura sockaddr\_in**

```
int
char IPstr[15] "10.0.0.1"
```

© 2002-2015 Volhys Bernal 17

## Chamada send()

© 2002-2015 Volhys Bernal 18

## Chamada send()

- **Função para transmissão de dados**
- **Podem ser utilizadas por clientes e servidores**

```
int send(int sd, void * txbuffer, int msgsize, int flags)
```

© 2002-2015 Volhys Bernal 19

### Chamada send()

❑ Exemplo:

```
char txbuffer[80];
. . .
status = send(sd, txbuffer, strlen(txbuffer)+1, 0)
if (status < 0)
    perror("Erro na chamada send");
. . .
```

© 2002-2015 Volhys Bernal 20

### Chamada recv()



© 2002-2015 Volhys Bernal 21

### Chamada recv()

❑ Recebimento de datagramas  
❑ Pode ser utilizada por clientes e servidores

```
int recv(int sd, void * rxbuffer, int rxbuffersize, int flags)
```

Socket  
Descriptor

Tamanho do  
buffer

Opções

Ponteiro para o buffer  
(end. do buffer de recepção)

© 2002-2015 Volhys Bernal 22

### Chamada recv()

❑ Exemplo:

```
char rxbuffer[80];
. . .
status = recv(sd, rxbuffer, sizeof(rxbuffer), 0)
if (status < 0)
    perror("Erro na chamada recv");
printf("MSG recebida: %s\n", rxbuffer);
. . .
```

© 2002-2015 Volhys Bernal 23

### Chamada recv()

❑ **Bloqueante**

- ❖ Se não existirem mensagens na fila de recepção o processo fica aguardando sua chegada
- ❖ Exceção: quando o socket for criado como não bloqueante (ver fcntl(2)).

❑ **Retorno**

- ❖ Se a chamada tiver sucesso, o valor retornado é o tamanho do datagrama

© 2002-2015 Volhys Bernal 24

### Chamada close()



© 2002-2015 Volhys Bernal 25

---

### Chamada close()

- ❑ **Objetivo**
  - ❖ Fechar o descritor de arquivos (neste caso, fecha o socket).
  - ❖ Se ainda existirem dados para serem transmitidos pelo socket, aguarda por alguns segundos a finalização desta transmissão.
- ❑ **Resultado**
  - ❖ Fecha o descritor do arquivo.
- ❑ **Sintaxe**

```
int close (int sd)
```

© 2002-2015 Volhys Bernal 26

---

### Chamada close()

- ❑ **Exemplo:**

```
int sd; // socket descriptor
. . .

status = close(sd);
if (status == -1)
    perror("Erro na chamada close");
. . .
```

© 2002-2015 Volhys Bernal 27

---

### Exercício



© 2002-2015 Volhys Bernal 28

---

### Exercício

- (1) Identifique a porta utilizada no serviço "echo".
  
- (2) Implemente um cliente para o serviço "echo" utilizando o protocolo UDP.
  - ❖ O serviço echo responde exatamente com a seqüência ASCII recebida.

© 2002-2015 Volhys Bernal 29

---

### Exercício

- (3) Identifique a porta utilizada no serviço "daytime".
  
- (4) Implemente um cliente para o serviço daytime utilizando o protocolo UDP.
  - ❖ O serviço daytime UDP responde com a data e hora do servidor no instante de recebimento do datagrama UDP.

© 2002-2015 Volhys Bernal 30

---

### Referências Bibliográficas



## Referências Bibliográficas

- **COMMER, DOUGLAS; STEVENS, DAVID**
  - ❖ Internetworking with TCP/IP: volume 3: client-server programming and applications
  - ❖ Prentice Hall
  - ❖ 1993